



PUC
CAMPINAS
PONTIFÍCIA UNIVERSIDADE CATÓLICA

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
Paulo Mangabeira Biocchi		RA: 16148363

24/10/2019

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.

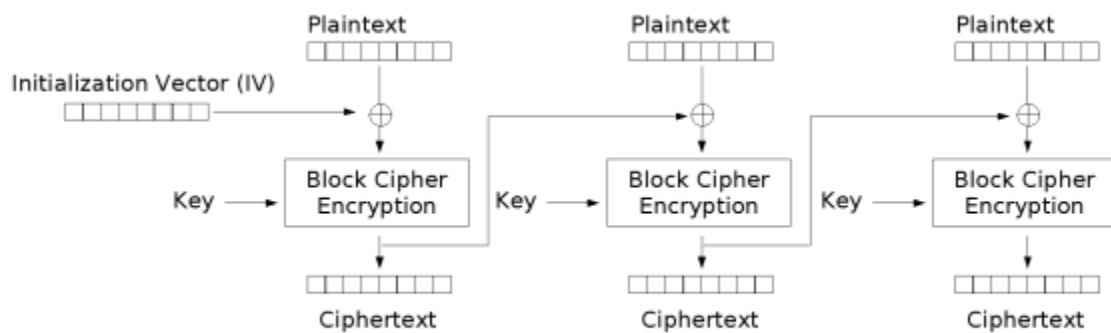
Para isto, foi desenvolvido um módulo de kernel Linux versão de um dispositivo criptográfico, capaz de cifrar e decifrar mensagens através do algoritmo AES em modo CBC, utilizando-se de uma chave e um vetor de inicialização fornecidos na inserção do módulo. O dispositivo também deveria ser capaz de calcular 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 4.4.190 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 possui 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.

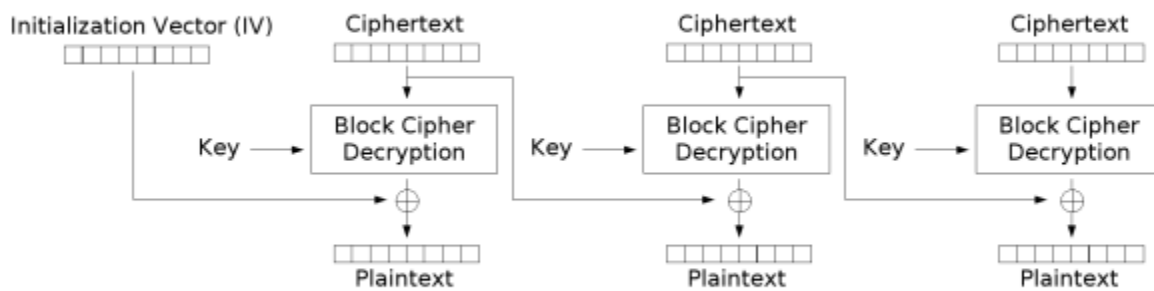
O algoritmo do AES CBC, na operação de cifragem, utiliza-se da chave e do valor inicial (representados em hexadecimal) fornecidos durante a carga do módulo, retornando uma string cifrada correspondente aos dados fornecidos durante a requisição. A etapa de cifragem é realizada recursivamente com os novos dados gerados, até se obter a palavra cifrada final.



Cipher Block Chaining (CBC) mode encryption

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

Já na operação de decifragem, o processo inverso da cifragem é realizado, fornecendo como resposta uma string correspondendo aos dados decifrados fornecidos durante a escrita no dispositivo, utilizando-se também da chave e do valor inicial fornecidos durante a carga do módulo.



Cipher Block Chaining (CBC) mode decryption

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

Para os validar a cifragem e a decifragem realizadas, foi utilizado o site <http://aes.online-domain-tools.com/>, cujo algoritmo de padding é o mesmo utilizado no projeto, onde inserimos "0" a direita da palavra até que o bloco seja completo.

Para a operação de cálculo do hash foi utilizado o algoritmo SHA1, onde é retornada uma string correspondendo ao resumo criptográfico em hexadecimal dos dados fornecidos durante a escrita no dispositivo. Para testes de validação, foi utilizado o site <http://www.sha1-online.com/>.

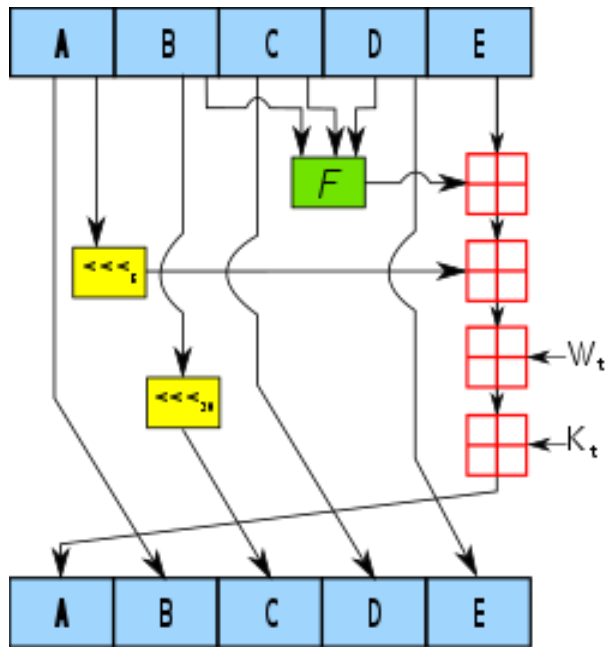


Figura 3: Diagrama do algoritmo de hash SHA-1.

Funcionamento do SHA1:

Uma iteração dentro da função de compressão do algoritmo:

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

F é uma função não-linear que varia;

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

W_t é a palavra da mensagem expandida da rodada t;

K_t é a constante da rodada t;

\boxplus denota uma adição módulo 232.

Para o projeto, também foram implementadas no módulo de kernel MUTEX LOCKS, utilizados para bloquear o programa de testes em espaço de usuário caso o dispositivo /dev/crypto já esteja sendo utilizado por outro processo.

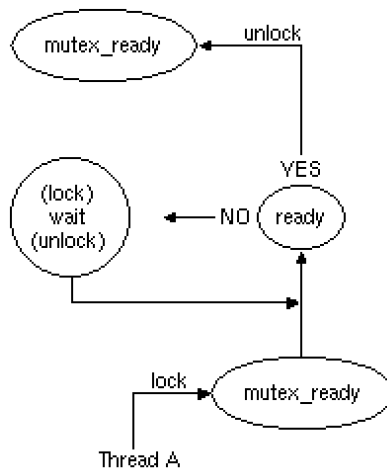


Figura 4: Diagrama do algoritmo do MUTEX LOCKS.

Por fim, o programa de testes abre o dispositivo do modulo, lê a entrada do usuário e escreve os dados no dispositivo /dev/crypto os dados digitados pelo usuário. Em seguida, o programa lê os dados retornados pelo modulo e os imprime no terminal.

Quanto a execução do projeto, o carregamento do modulo é executado no seguinte formato:

```
insmod cryptochar.ko key="0102030405060708A1A2A3A4A5A6A7A8" iv="0102030405060708A1A2A3A4A5A6A7A8"
```

No programa teste, o primeiro caractere define a operação, sendo estas cifrar (c), decifrar (d) e calcular o hash (h) do dado informado.

Os dados de chave e valor de início são passados em formato de string, representados em hexadecimal (cada byte corresponde a dois dígitos hexa), enquanto o dado a sofrer a operação é passado como plaintext.

3.Resultados

Primeiramente, foi realizado a compilação do modulo e do programa de testes utilizando um arquivo makefile.

```
user@ubuntuvm:~/shared/Projeto1$ make
make -C /lib/modules/4.4.190-marc0/build/ M=/home/user/shared/Projeto1 modules
make[1]: Entering directory '/home/user/linux-4.4.190'
CC [M] /home/user/shared/Projeto1/cryptochar.o
/home/user/shared/Projeto1/cryptochar.c: In function 'cryptoapi_init':
/home/user/shared/Projeto1/cryptochar.c:293:5: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-statement]
message * INPUT = alloc_with_padding(input, KEY_SIZE);
^
/home/user/shared/Projeto1/cryptochar.c:298:5: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-statement]
char * txt = sg_virt(&sk.out);
^
/home/user/shared/Projeto1/cryptochar.c: In function 'decryptapi_init':
/home/user/shared/Projeto1/cryptochar.c:331:5: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-statement]
message INPUT;
^
/home/user/shared/Projeto1/cryptochar.c:341:5: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-statement]
char * txt = sg_virt(&sk.out);
^
/home/user/shared/Projeto1/cryptochar.c: In function 'cryptosha1_init':
/home/user/shared/Projeto1/cryptochar.c:391:5: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-statement]
int i;
^
In file included from include/linux/printk.h:6:0,
                 from include/linux/kernel.h:13,
                 from include/linux/crypto.h:21,
                 from include/crypto/algapi.h:15,
                 from include/crypto/internal/skcipher.h:16,
                 from /home/user/shared/Projeto1/cryptochar.c:2:
include/linux/kern_levels.h:4:18: warning: format '%d' expects argument of type 'int', but argument 2 has type 'size_t {aka long unsigned int}' [-Wformat=]
#define KERN_SOH "\001" /* ASCII Start Of Header */
^
include/linux/kern_levels.h:13:19: note: in expansion of macro 'KERN_SOH'
#define KERN_INFO KERN_SOH "6" /* informational */
^
include/linux/printk.h:259:9: note: in expansion of macro 'KERN_INFO'
printk(KERN_INFO pr_fmt(fmt), ##__VA_ARGS__)
^
/home/user/shared/Projeto1/cryptochar.c:502:5: note: in expansion of macro 'pr_info'
pr_info("size_msg = %d", strlen(msg));
^
/home/user/shared/Projeto1/cryptochar.c:504:5: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-statement]
char * x = kmalloc(strlen(msg) - 1, GFP_KERNEL);
^
/home/user/shared/Projeto1/cryptochar.c: In function 'dev_read':
/home/user/shared/Projeto1/cryptochar.c:494:1: warning: control reaches end of non-void function [-Wreturn-type]
}
^
Building modules, stage 2.
make[2]: Warning: File '/home/user/shared/Projeto1/cryptochar.o' has modification time 1.5 s in the future
```

Figura 5: Output do comando "make"

Em seguida, o modulo foi inserido com os valores de key = "0102030405060708A1A2A3A4A5A6A7A8" e iv = "0102030405060708A1A2A3A4A5A6A7A8".

```
user@ubuntuvm:~/shared/Projeto1$ sudo insmod cryptochar.ko key="0102030405060708A1A2A3A4A5A6A7A8" iv="0102030405060708A1A2A3A4A5A6A7A8"
```

Figura 6: Output do Makefile passando os valores de iv e key.

Os dados foram verificados através do comando *dmesg*.

```
[ 1231.338782] CRYPTOChar: Initializing the CRYPTOChar LKM
[ 1231.338787] key = 0102030405060708A1A2A3A4A5A6A7A8
[ 1231.338821] initialization vector = 0102030405060708A1A2A3A4A5A6A7A8
[ 1231.338858] CRYPTOChar: registered correctly with major number 247
[ 1231.338879] CRYPTOChar: device class registered correctly
[ 1231.340639] CRYPTOChar: device class created correctly
user@ubuntuv:~/shared/Projeto1$
```

Figura 7: Mensagens do kernel em sua inserção.

Em seguida, foi executado o programa teste. Primeiramente, foi testada a cifragem de dados.

```
user@ubuntuv:~/shared/Projeto1$ sudo bash
root@ubuntuv:~/shared/Projeto1# ./teste_userspace
Type in a short string to send to the kernel module:
c Testando
Writing message to the device [c Testando].
Press ENTER to read back from the device...

Reading from the device...
SIZE=16
The received message is:
msg[0] = 0x7C = |
msg[1] = 0x4F = O
msg[2] = 0x80 = ♦
msg[3] = 0x4E = N
msg[4] = 0xB7 = ♦
msg[5] = 0x46 = F
msg[6] = 0xBC = ♦
msg[7] = 0xFF = ♦
msg[8] = 0x99 = ♦
msg[9] = 0x2E = .
msg[10] = 0xF7 = ♦
msg[11] = 0x81 = ♦
msg[12] = 0x6C = l
msg[13] = 0x85 = ♦
msg[14] = 0xD0 = ♦
msg[15] = 0xB8 = ♦
```

Figura 8: Entrada de dados e Output da operação de cifra do módulo.

Em seguida, foram verificadas as mensagens de retorno do modulo para o kernel.

```
[ 524.596688] CRYPTOChar: Device has been opened
[ 550.440227] msg = c Testando
[ 550.440235] op = c
[ 550.440238] size_msg = 10
[ 550.440244] dados=Testando
[ 550.440251] PADDING EXECUTADO!
[ 550.447191] ENCRYPT AES128 OPERATION:
[ 550.447201] Encryption request successful
[ 550.447206] sg[0]= 0x7C
[ 550.447210] sg[1]= 0x4F
[ 550.447213] sg[2]= 0x80
[ 550.447215] sg[3]= 0x4E
[ 550.447218] sg[4]= 0xB7
[ 550.447221] sg[5]= 0x46
[ 550.447224] sg[6]= 0xBC
[ 550.447226] sg[7]= 0xFF
[ 550.447229] sg[8]= 0x99
[ 550.447232] sg[9]= 0x2E
[ 550.447235] sg[10]= 0xF7
[ 550.447237] sg[11]= 0x81
[ 550.447240] sg[12]= 0x6C
[ 550.447243] sg[13]= 0x85
[ 550.447246] sg[14]= 0xD0
[ 550.447248] sg[15]= 0xB8
[ 550.447254] CRYPTOChar: WRITE OK
[ 552.526406] CRYPTOChar: Sent 16 characters to the user
[ 552.526606] CRYPTOChar: Device successfully closed
```

Figura 9: Mensagens do modulo para a criptografia.

O resultado obtido foi comparado com um algoritmo da internet.

aes.online-domain-tools.com

Input type: Text

Input text: (plain) Testando

Plaintext Hex Autodetect: ON | OFF

Function: AES

Mode: CBC (cipher block chaining)

Key: (hex) 0102030405060708A1A2A3A4A5A6A7A8

Plaintext Hex

Init. vector: 01 02 03 04 05 06 07 08 A1 A2 A3 A4 A5 A6 A7 A8

> Encrypt! > Decrypt!

Initialization vector: 0102030405060708a1a2a3a4a5a6a7a8 (256 bits)

Encrypted text: 7c 4f 80 4e b7 46 bc ff 99 2e f7 81 6c 85 d0 b8 | 0 . N . F ½ ÿ . . ÷ . l Ð ,

[Download as a binary file] [?] Inactive

Figura 10: Comparação de resultados de cifragem no site <http://aes.online-domain-tools.com/>

Com isso, foi verificado o correto funcionamento da cifragem.

Em seguida, a decifragem foi testada.

```
root@ubuntuvm:~/shared/Projeto1# ./teste_userspace
Type in a short string to send to the kernel module:
d 7C4F804EB746BCFF992EF7816C85D0B8
Writing message to the device [d 7C4F804EB746BCFF992EF7816C85D0B8].
Press ENTER to read back from the device...

Reading from the device...
SIZE=16
The received message is:
msg[0] = 0x54 = T
msg[1] = 0x65 = e
msg[2] = 0x73 = s
msg[3] = 0x74 = t
msg[4] = 0x61 = a
msg[5] = 0x6E = n
msg[6] = 0x64 = d
msg[7] = 0x6F = o
msg[8] = 0x0 =
msg[9] = 0x0 =
msg[10] = 0x0 =
msg[11] = 0x0 =
msg[12] = 0x0 =
msg[13] = 0x0 =
msg[14] = 0x0 =
msg[15] = 0x0 =
root@ubuntuvm:~/shared/Projeto1#
```

Figura 11: Entrada de dados e Output da operação de decifragem do módulo.

Após o pedido de decifragem, foram verificadas as mensagens do modulo para o kernel.

```
593.484452] CRYPTOChar: Device has been opened
616.871590] msg = d 7C4F804EB746BCFF992EF7816C85D0B8
616.871598] op = d
616.871601] size_msg = 34
616.871607] dados=7C4F804EB746BCFF992EF7816C85D0B8
616.871631] DECRYPT AES128 OPERATION:
616.871636] Encryption request successful
616.871640] sg[0]= 0x54
616.871642] sg[1]= 0x65
616.871645] sg[2]= 0x73
616.871647] sg[3]= 0x74
616.871650] sg[4]= 0x61
616.871652] sg[5]= 0x6E
616.871655] sg[6]= 0x64
616.871657] sg[7]= 0x6F
616.871660] sg[8]= 0x00
616.871662] sg[9]= 0x00
616.871664] sg[10]= 0x00
616.871667] sg[11]= 0x00
616.871669] sg[12]= 0x00
616.871672] sg[13]= 0x00
616.871674] sg[14]= 0x00
616.871677] sg[15]= 0x00
616.871682] CRYPTOChar: WRITE OK
618.745539] CRYPTOChar: Sent 16 characters to the user
618.745861]
CRYPTOChar: Device successfully closed
root@ubuntuvn:~/shared/Projeto1#
```

Figura 12: Mensagens do modulo para a decifragem.

O resultado obtido foi comparado com um algoritmo da internet.

Input type:

Input text:
(hex)

☐ Plaintext ☒ Hex Autodetect: **ON** | OFF

Function:

Mode:

Key:
(hex)

☐ Plaintext ☒ Hex

Init. vector:

Initialization vector:
0102030405060708a1a2a3a4a5a6a7a8 (256 bits)

Decrypted text:
 54 65 73 74 61 6e 64 6f 00 00 00 00 00 00 00 00 | T e s t a n d o
[\[Download as a binary file\] \[?\]](#) Inactive

Figura 13: Comparação de resultados de decifragem no site <http://aes.online-domain-tools.com/>

Com isso, foi verificado o correto funcionamento da cifragem.

Em seguida, o cálculo do hash foi testado.

```
root@ubuntuvm:~/shared/Projeto1# ./teste_userspace
Type in a short string to send to the kernel module:
h Testando123EsseProjetoEhUmProjetoDeSOB_2019
Writing message to the device [h Testando123EsseProjetoEhUmProjetoDeSOB_2019].
Press ENTER to read back from the device...

Reading from the device...
SIZE=20
The received message is:
msg[0] = 0x7A = z
msg[1] = 0xA4 = ♦
msg[2] = 0x60 = `
msg[3] = 0xC6 = ♦
msg[4] = 0x48 = H
msg[5] = 0x84 = ♦
msg[6] = 0xD6 = ♦
msg[7] = 0xF6 = ♦
msg[8] = 0x88 = ♦
msg[9] = 0xEB = ♦
msg[10] = 0xFC = ♦
msg[11] = 0xF4 = ♦
msg[12] = 0x5 = 
msg[13] = 0x21 = !
msg[14] = 0x8E = ♦
msg[15] = 0xA6 = ♦
msg[16] = 0x79 = y
msg[17] = 0x93 = ♦
msg[18] = 0x25 = %
msg[19] = 0x61 = a
root@ubuntuvm:~/shared/Projeto1#
```

Figura 14: Entrada de dados e Output da operação de hash do módulo.

Após o pedido de decifragem, foram verificadas as mensagens do modulo para o kernel.

```
[ 958.062781] CRYPTOChar: Device has been opened
[ 963.168571] msg = h Testando123EsseProjetoEhUmProjetoDeSOB_2019
[ 963.168579] op = h
[ 963.168581] size_msg = 45
[ 963.168599] dados=Testando123EsseProjetoEhUmProjetoDeSOB_2019
[ 963.168632] HASH[0]= 0x7A
[ 963.168635] HASH[1]= 0xA4
[ 963.168637] HASH[2]= 0x60
[ 963.168640] HASH[3]= 0xC6
[ 963.168642] HASH[4]= 0x48
[ 963.168645] HASH[5]= 0x84
[ 963.168647] HASH[6]= 0xD6
[ 963.168650] HASH[7]= 0xF6
[ 963.168652] HASH[8]= 0x88
[ 963.168654] HASH[9]= 0xEB
[ 963.168657] HASH[10]= 0xFC
[ 963.168660] HASH[11]= 0xF4
[ 963.168662] HASH[12]= 0x05
[ 963.168665] HASH[13]= 0x21
[ 963.168667] HASH[14]= 0x8E
[ 963.168670] HASH[15]= 0xA6
[ 963.168672] HASH[16]= 0x79
[ 963.168675] HASH[17]= 0x93
[ 963.168677] HASH[18]= 0x25
[ 963.168679] HASH[19]= 0x61
[ 963.168682] CRYPTOChar: WRITE OK
[ 964.914273] CRYPTOChar: Sent 20 characters to the user
[ 964.914630] CRYPTOChar: Device successfully closed
root@ubuntuvm:~/shared/Projeto1#
```

Figura 15: Mensagens do modulo para o hash.

O resultado obtido foi comparado com um algoritmo da internet.

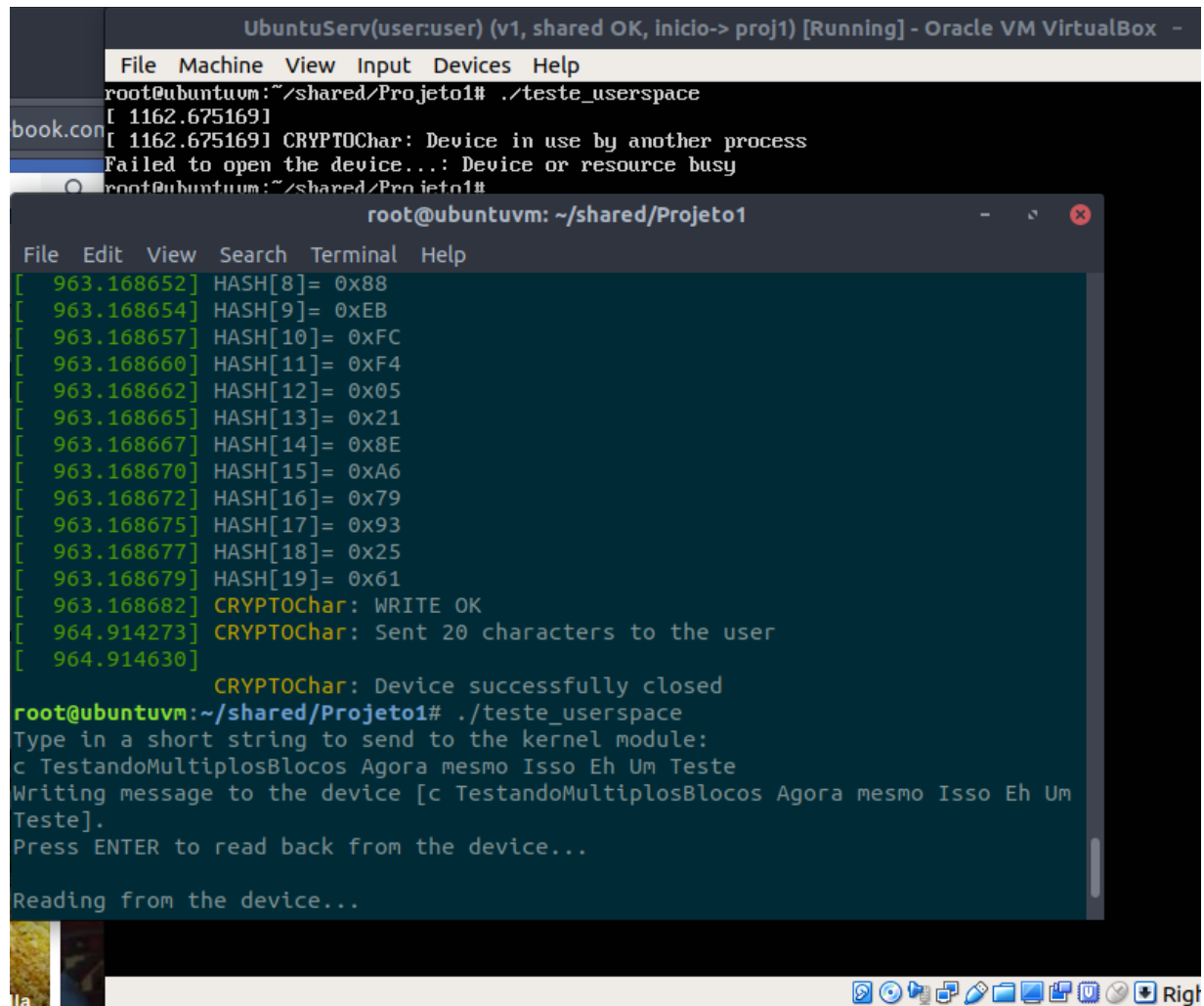


The screenshot shows a web browser window with the address bar displaying "www.sha1-online.com". The page content includes a navigation bar with links: "Home Page | [SHA1 in JAVA](#) | [Secure password generator](#) | [Linux](#)". The main heading is "SHA1 and other hash functions online generator". Below this, there is a text input field containing the string "Testando123EsseProjetoEhUmProjetoDeSOB_2019" and a button labeled "hash". Underneath the input field is a dropdown menu currently set to "sha-1". The result is displayed as "Result for sha1: 7aa460c64884d6f688ebfcf405218ea679932561".

Figura 16: Comparação de resultados de hash no site <http://www.sha1-online.com/>

Com isso, foi verificado o correto funcionamento do hash.

Em seguida, foram testados os mutex locks.



```
UbuntuServ(user:user) (v1, shared OK, inicio-> proj1) [Running] - Oracle VM VirtualBox -
File Machine View Input Devices Help
root@ubuntuvvm:~/shared/Projeto1# ./teste_userspace
[ 1162.675169]
[ 1162.675169] CRYPTOChar: Device in use by another process
Failed to open the device...: Device or resource busy
root@ubuntuvvm:~/shared/Projeto1#

root@ubuntuvvm: ~/shared/Projeto1
File Edit View Search Terminal Help
[ 963.168652] HASH[8]= 0x88
[ 963.168654] HASH[9]= 0xEB
[ 963.168657] HASH[10]= 0xFC
[ 963.168660] HASH[11]= 0xF4
[ 963.168662] HASH[12]= 0x05
[ 963.168665] HASH[13]= 0x21
[ 963.168667] HASH[14]= 0x8E
[ 963.168670] HASH[15]= 0xA6
[ 963.168672] HASH[16]= 0x79
[ 963.168675] HASH[17]= 0x93
[ 963.168677] HASH[18]= 0x25
[ 963.168679] HASH[19]= 0x61
[ 963.168682] CRYPTOChar: WRITE OK
[ 964.914273] CRYPTOChar: Sent 20 characters to the user
[ 964.914630]
CRYPTOChar: Device successfully closed
root@ubuntuvvm:~/shared/Projeto1# ./teste_userspace
Type in a short string to send to the kernel module:
c TestandoMultiplosBlocos Agora mesmo Isso Eh Um Teste
Writing message to the device [c TestandoMultiplosBlocos Agora mesmo Isso Eh Um
Teste].
Press ENTER to read back from the device...

Reading from the device...
```

Figura 17: teste do MUTEX LOCKS no módulo.

Após o acesso dos 2 programas de teste, foram verificadas as mensagens do modulo para o kernel.



```
[ 1162.675169]
CRYPTOChar: Device in use by another process
```

Figura 18: Mensagens do modulo para múltiplos acessos.

Com isso, foi verificado o correto funcionamento do mutex.

4. Conclusão

O projeto funcionou como esperado, apresentando o correto funcionamento de suas funções.

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

Através do estudo deste módulo, foram aprimorados os conceitos de implementação, compilação, instalação e testes de um novo módulo de kernel que realiza as funções de cifrar, decifrar e calcular o hash dos dados fornecidos pelo usuário. Também aprendeu-se sobre a utilização de outras funções do kernel, como scatterlists e mutexs.