

Pontifícia Universidade Católica de Campinas Faculdade de Engenharia de Computação - FECOMP

Sistemas Operacionais B – Relatório Experimento 2

Chamada de sistema criptográfica no kernel Linux

Beatriz Morelatto Lorente RA: 18071597

Cesar Marrote Manzano RA: 18051755

Fabrício Silva Cardoso RA: 18023481

Pedro Ignácio Trevisan RA: 18016568

**Sumário**

1. **Introdução.....................................................................................................3**
2. **Algoritmo de criptografia utilizado.............................................................4**
3. **Fluxo das informações................................................................................5**
4. **Passos para implementação de uma syscall...........................................6**
5. **Resultados....................................................................................................8**
6. **Conclusão...................................................................................................11**

**Introdução**

O experimento desenvolvido pretende demonstrar os passos feitos para o desenvolvimento de chamadas de sistema (system calls) que podem armazenar e ler arquivos usando a API criptográfica do kernel. Foram feitas duas chamadas de sistema:

* Na primeira system call, o usuário envia uma mensagem que é criptografada e depois gravada no arquivo
* Na segunda system call, a mensagem armazenada no arquivo é lida, decifrada e enviada para o usuário.

Para a cifragem e decifragem da mensagem foi usado o algoritmo AES em modo ECB.

**Algoritmos de criptografia utilizados**

Como citado anteriormente, para cifrar e decifrar a string fornecida pelo usuário foi utilizado o algoritmo AES em modo ECB.

Primeiramente é necessário entender como o algoritmo AES (Advanced Encryption Standard) funciona. O algoritmo consiste em uma criptografia simétrica de blocos de tamanho fixo (utilizamos blocos de 16 bytes, 128 bits). O algoritmo usa uma chave para cifrar e decifrar os blocos e esta também tem 16 bytes.

O modo de criptografia ECB (Eletronic CodeBook), é um dos mais simples que temos, é a primeira geração dos modos AES. A mensagem é dividida em blocos que são critografados separadamente. Como o modo é bem simples, ele não oferece muita segurança, uma vez que não oculta padrões de dados, não sendo recomendado caso o sistema necessite de uma confiabilidade alta. O esquema abaixo mostra como o algoritmo funciona.

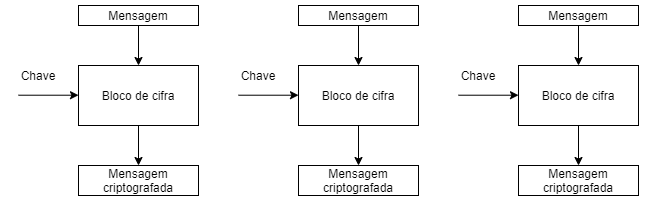


Figura 1 - Funcionamento do algoritmo AES em modo ECB

**Fluxo das informações**

Para entender como o programa funciona, será mostrado um esquema, em alto nível, mostrando o fluxo das informações. Esse esquema engloba tanto o programa de teste como as chamadas de sistema criadas.

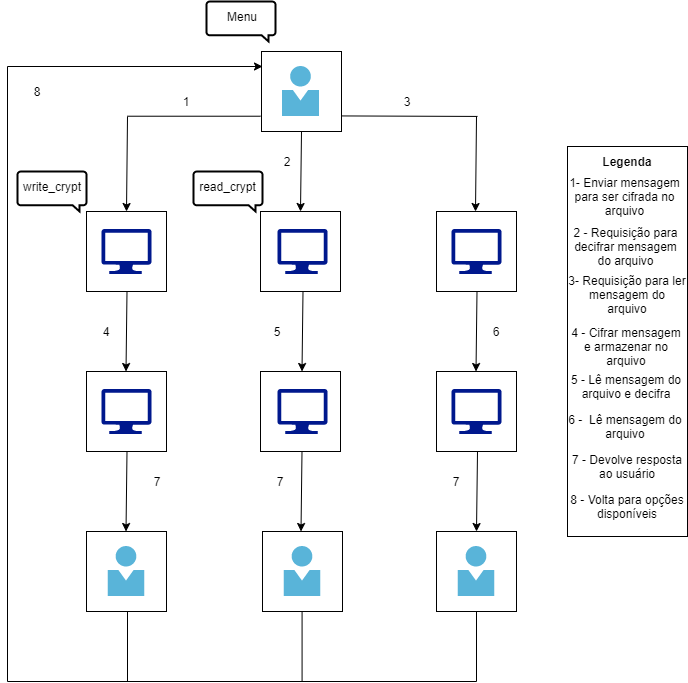


Figura 2 - Fluxo das informações no programa

**Passos para implementação de uma syscall**

Para implementar uma syscall há alguns passos principais que serão mostrados abaixo.

Para a nossa implementação foi usada a versão do linux 4.15.0. Para instalar essa versão ou outras disponíveis, basta digitar ‘sudo apt-cache search linux-source’, no terminal e depois ‘sudo apt-get install linux-source-[versão deejada]’.

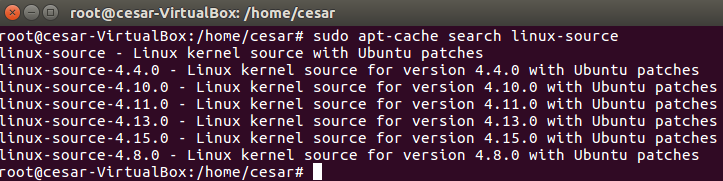


Figura 3 - Versões de kernel disponíveis após rodar o comando ‘sudo apt-cache search linux-source’

Depois de instalar a versão desejada do kernel do linux, ela estará disponível na pasta ‘/usr/src’.

Acesse a pasta da sua versão do kernel e crie um novo diretório, que conterá todos os arquivos da sua syscall. Para criar o diretório, digite ‘mkdir [nome do diretório]’ e depois entre com o comando ‘cd [nome do diretório]’.

Após criar um ou mais arquivos que conterá sua(s) syscall(s), é necessário criar um Makefile dentro do diretório, contendo o conteúdo ‘obj-y:=[nome do arquivo].o’. Esse arquivo será responsável por compilar sua chamada de sistema.

Com o Makefile criado é necessário modificar outro Makefile, mas desta vez do kernel. Volte para a pasta principal do kernel e edite-o. Procure pelo nome ‘core-y’ e adicione o nome do diretório criado no final da linha, como mostrado abaixo.

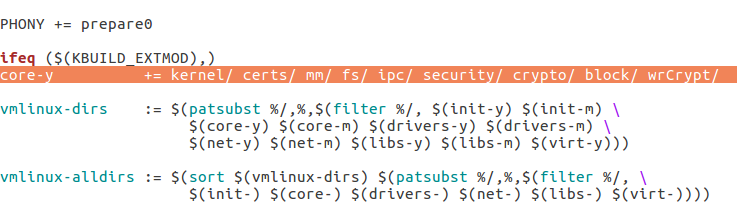


Figura 4 - Modificação do Makefile do kernel

Depois é necessário modificar o arquivo ‘syscall\_64.tbl’, encontrado no caminho ‘/arch/x86/entry/syscalls’. Depois adicione sua(s) syscall(s), como mostrado abaixo.

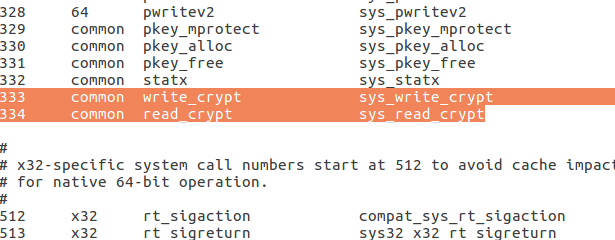


Figura 5 - Modificação syscall\_64.tbl

Por fim é necessário o arquivo ’syscalls.h’, disponível no diretório ‘/include/linux’. Aqui será feita a declaração do corpor, assinatura, da função, no final do arquivo.

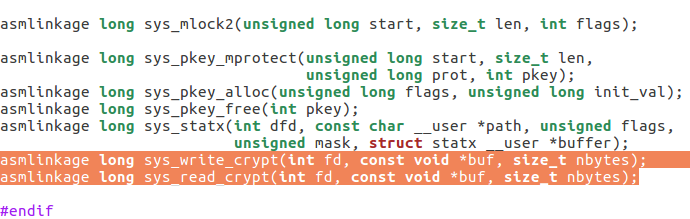


Figura 6 - Modificação do arquivo 'syscalls.h'

Com tudo pronto é necessário compilar o seu kernel, para que suas modificações sejam salvas. Para isso é necessário rodar inicialmente o comando ‘make menuconfig’ para modificar o que você quiser no kernel. É importante lembrar que é interessante ter um kernel que compile rapidamente, uma vez que toda alteração que for feita na syscall, será necessário compilar o kernel. Depois de configurar o kernel, rode o comando ‘make -jN && make modules install -jN && make install -jN && reboot’, sendo N o número de CPUs disponíveis na sua máquina virtual. Após esse processo, suas modificações poderão ser testadas.

**Resultados**

Para a apresentação dos resultados obtidos, primeiramente será mostrado o resultado da escrita no arquivo e depois o resultado da leitura do mesmo. Para testar a chamada de sistema, foi utilizado um gerador de palavras aleatórias., que gerando quatro palavras e também escolhemos uma para testar, que são:

* Caderno (1)
* Calcanhar (2)
* Mirante (3)
* Gafanhoto (4)
* Ordem e progresso (5)

**1 - Primeira Palavra**

*Escrita no arquivo*



Figura 7 - Entrada da palavra ‘Caderno’



Figura 8 – Resultado da execução da ‘syscall write’



Figura 9 – A string de entrada cifrada

*Leitura do arquivo*

Figura 10 – String descriptografada

Figura 11– Resultado da execução da ‘syscall read’

**2 - Segunda Palavra**

*Escrita no arquivo*



Figura 12- Entrada da palavra 'Calcanhar’



Figura 13– Resultado da execução da ‘syscall write’



Figura 14– A string de entrada cifrada

*Leitura do arquivo*



Figura 15– String descriptografada



Figura 16– Resultado da execução da ‘syscall read’

**3 - Terceira Palavra**

*Escrita no arquivo*



Figura 17- Entrada da palavra 'Mirante’



Figura 18– Resultado da execução da ‘syscall write’



Figura 19– A string de entrada cifrada

*Leitura do arquivo*



Figura 20– String descriptografada



Figura 21– Resultado da execução da ‘syscall read’

**4 - Quarta Palavra**

*Escrita no arquivo*



Figura 22- Entrada da palavra 'Gafanhoto’



Figura 23– Resultado da execução da ‘syscall write’



Figura 24– A string de entrada cifrada

*Leitura do arquivo*



Figura 25– String descriptografada



Figura 26– Resultado da execução da ‘syscall read’

**5 - Quinta Palavra**

*Escrita no arquivo*



Figura 27- Entrada da palavra 'Ordem e progresso’



Figura 28– Resultado da execução da ‘syscall write’



Figura 29– A string de entrada cifrada

*Leitura do arquivo*



Figura 30– String descriptografada



Figura 31– Resultado da execução da ‘syscall read’

**Conclusão**

Com o experimento foi possível compreender melhor como uma syscall funciona no ambiente Linux. E ainda implementar novas chamadas de sistemas que fazem uso da API de criptografia do kernel, abordando todo o conhecimento adquirido no semestre.