Trabalho de Software Básico - Turma 2010-1	
Nome: Francisco Panis Kaseker	GRR20071909
Título: Explicação e implementação de uma SYSCALL	Data: 30/06/2010

1 Introdução

Basicamente uma *SYSCALL* é uma chamada da aplicação feito ao *Kernel* do sistema operacional. Essa chamada executa alguma rotina no sistema, mas dentro da camada do *Kernel*. A segunda seção explicará com maior profundidade do que tratam as *SYSCALLs* (história, exemplos etc) e, por fim, a terceira tratará de explicar como adicionar uma chamada ao sistema no Linux.

2 SYSCALLs: O que são

2.1 História

Devido a necessidade de alguns aplicativos precisarem obter determinados ações ou resultados do sistema operacional sem para isso terem de rodar sob os privilégios do modo *root*, as *SYSCALLs* vieram para suprir essa necessidade sem para isso diminuir a segurança do sistema operacional.

2.2 Porque usá-las

Essencialmente para garantir segurança ao sistema ao não permitir que os aplicativos tenham que rodar sob os privilégios do *root*, mas que possam desempenhar suas funções no sistema normalmente.

2.3 Exemplos

Nos sistemas *UNIX-like* nós temos a *POSIX* que desempanhm diversas funções (como, por exemplo, open, read, write, close, wait, exec, fork, exit kill etc). Nos sistemas baseados em Windows, temos a *WIN32*, somente implementadas pela Microsoft. Parte delas está descrita na documentação da Microsoft e outra parte não.

2.4 Implementaçõs mais comuns

As chamadas ao sistema possuem diversas utilidades atualmente. Na parte de controle de processos, por exemplo, temos a criação ou fechamento de um processo ou alocação e liberação de memória. Para o gerenciamento de arquivos, temos a abertura, fechamento, escrita ou leitura de arquivos. Há ainda chamadas ao sistema para envio de dados na rede, gerenciamento de drivers (hd, cd, dvd etc) e tantas outras funções cujo controle é somente do sistema operacional ou do *root* .

2.5 O modelo POSIX

O modelo *POSIX* é uma padronização de chamadas do sistema para tentar garantir a portabilidade de um código em diferentes sistemas operacionais. O modelo *POSIX* é seguido pelos sistemas Unix-like, porém isso não é uma regra. O sistema *FreeBSD*, por exemplo, tem mais que o dobro de *SYSCALLs* que o definido pela *POSIX*.

2.6 O modelo WIN32

Essa API é desenvolvida para o sistema operacional Windows, da Microsoft. Ela abrange desde os sistemas 16bits até os 64bits do seu portfólio. Embora a WIN32 seja um modelo para fornecer recursos do sistema operacional aos aplicativos, ela permite que outras API's tenham acesso diretamente aos recursos de hardware do sistema, especialmente para hardware gráfico (através

da win32k.sys) e de rede.

3 Implementando uma SYSCALL

Será implementado uma *SYSCALL* no *Kernel* Linux x86 que tem a função de exibir quantas vezes ela foi chamada. Será usado uma variável global como contador e a *SYSCALL* terá de somar o valor um a essa variável global sempre que for chamada. Todo printk ("printf do Kernel") será ligo no seguinte log: /var/log/syslog
A *SYSCALL* aqui implementada foi a de número 339.

3.1 Compilando o Kernel

Foi utilizado o tutorial abaixo para compilar o Kernel; Importante: O autor do artigo do site abaixo referenciado é o mesmo autor desse trabalho. http://gerencievocemesmo.com.br/site/?p=281

3.2 Implementando uma SYSCALL no Linux x86

1. Baixar o Kernel e extraí-lo em algum lugar; Usei o "/usr/src/".
2. Para facilitar, criei um link simbólico para usar o path "/usr/src/linux"como source do Kernel.
3. Crie um arquivo chamado "minhasys.c"em "/usr/src/kernel/minhasys/". O seu conteúdo será o seguinte:

```
#include <linux/linkage.h>
asmlinkage int sys_minhasys() {
MINHAVARIAVELGLOBAL++;
printk("CHAMADA %d VEZES!\n", MINHAVARIAVELGLOBAL);
return 0;
}
```

4. Agora vamos modificar alguns arquivos para tornar a nova *SYSCALL* disponível ao *Kernel*.

   - arquivo: /usr/src/linux/arch/x86/include/asm/unistd_32.h
     adicione na ultima linha: #define __NR_minhasys NUMERO
     Substitua NUMERO pelo último número do define da lista incrementado de "1".
   - arquivo: /usr/src/linux/arch/x86/include/asm/syscalls.h
     adicione ao fim da lista: asmlinkage int sys_helloworld();
   - arquivo: /usr/src/linux/arch/x86/kernel/syscall_table_32.c
     adicione: .long sys_minhasys
   - arquivo: /usr/src/linux/Makefile linha original: core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/
     nova linha: core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ minhasys/

5. Agora crie um arquivo chamado "Makefile"dentro do diretório "minhasys"com o seguinte conteúdoo:
   obj-y := minhasys.o

6. O contador (variável int MINHAVARIAVELGLOBAL) foi declarado dentro da syscall como static, assim a declaração será feita uma única vez e o valor não será perdido.

7. Agora é só compilar o *Kernel*.

4 Dicas adicionais

- O NUMERO de uma *SYSCALL* pode ser qualquer uma dentro do formato (int), desde que já não esteja sendo usada.

- No how-to acima a syscall foi colocada dentro de um arquivo separado, mas poderia ter sido incluída facilmente dentro do arquivo "kernel/sys.c", junto as outras *SYSCALL*; Fazendo isso, não é necessário criar ou modificar nenhum Makefile.

5 Referências

- http://macboypro.wordpress.com/2009/05/15/adding-a-custom-system-call-to-the-linux-os/
- http://gerencievocemesmo.com.br/site/?p=281
- http://www.linuxchix.org/content/courses/kernel_hacking/lesson8
- http://ubuntuforums.org/attachment.php?attachmentid=149947&d=1268472461
- http://en.wikipedia.org/wiki/System_call
- http://pt.wikipedia.org/wiki/POSIX
- http://en.wikipedia.org/wiki/Windows_API
- http://blog.veiga.eti.br/niveis-de-log-do-printk-para-depurar-o-kernel-pela-console/