Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

SSC0141 – Sistemas Operacionais II

Projeto 1 – Implementação de um Keylogger para Sistemas Linux
Parte 1 – Pesquisa e Proposta de Implementação

São Carlos, 30 de agosto de 2009

Trabalho desenvolvido pelos alunos:

Ubiratan Soares (5634292)
Ulisses Soares (5377365)
Vinicius Grippa (5119050)

Introdução

Keystroke logging – ou simplesmente *keylogging* – é a prática de conhecer os toques nas teclas de um teclado de computador, tipicamente de maneira que o usuário não sabia que suas ações estejam sendo monitoradas [1]. Apesar da viabilidade de uma implementação por hardware, **keyloggers** normalmente são construídos em software, de maneira a operar segundo características do sistema operacional do computador alvo.

A construção de um keylogger baseado em software pode seguir uma das seguintes estratégias[1]:

(a) **Baseados em Hypervisor** : o software keylogger teoricamente reside em um malware hypervisor, se comportando similarmente a uma máquina virtual.

(b) **Baseados em Kernel** : o software keylogger reside no kernel do computador alvo. A implementação frequente é através de *rootkits,* que subvertem o kernel do sistema operacional para ganhar acesso não-autorizado ao hardware da máquina, por exemplo, atuando como um driver de teclado. São mais difíceis de serem escritos, mas também detectados, em especial por aplicações de modo usuário.

(c) **Baseados em Hook** : o software keylogger usa funcionalidades oferecidas pelo sistema operacional para aplicações de maneira a subescrever os eventos do teclado de maneira legítima. O sistema operacional irá notificar o keylogger a cada vez que uma tecla for pressionada.

(d) **Métodos Passivos** : o software keylogger é implementado usando-se as APIs do sistema operacional para subescrever os eventos do teclado. São mais simples de escrever, porém podem levar a sobrecarga da CPU, uma vez que uma chamada a API será executada a cada toque.

A Jornada de um Caractere

Em um sistema Unix, ao se pressionar um tecla do teclado, o caractere não é simplesmente anexado aos buffers do gerenciador de terminal (**tty**) imediatamente após ser enviado por alguma porta do hardware [3]. Uma série de etapas anteriores são necessárias antes que o Kernel saiba qual é o caractere correspondente à tecla pressionada, e antes que esse esteja visível na tela em resposta à ação do usuário.

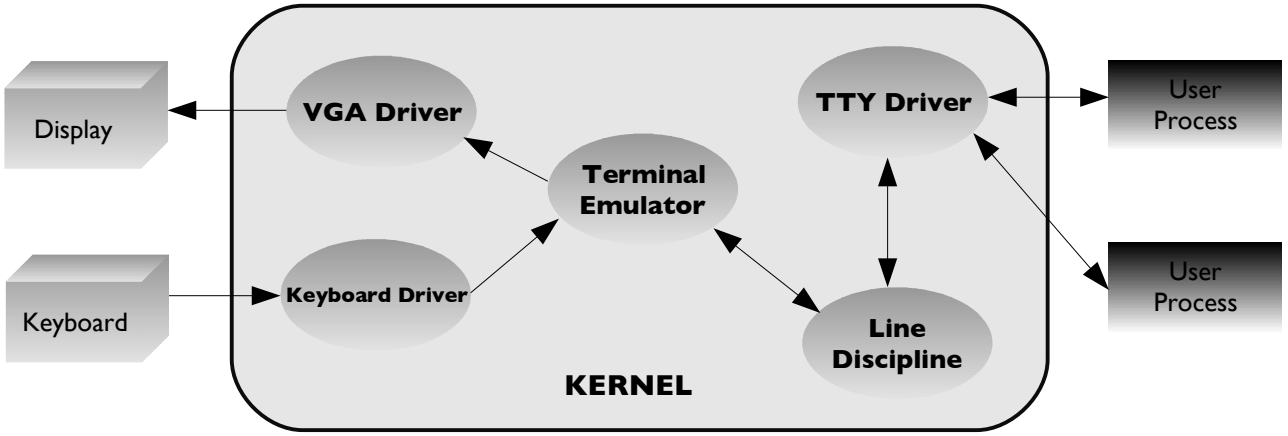
Inicialmente, há um circuito integrado, interno a qualquer teclado, responsável por verificar quando novas teclas são pressionadas ou despressionadas. Uma vez que essas ações são identificadas, essas informações são enviadas através de um ou mais bytes pelo cabo serial (para um teclado PS2 ou USB) para o computador. O chip no teclado pode escolher enviar esses dados em três condificações diferentes, dependendo dos eventos associados ao pressionar e liberar das teclas. Essas condificações são chamadas **scancodes**, sendo elas rotuladas de 1 a 3 [2]. O modo default para o envio de scancodes é o modo 2[3].

Para o barramento PS2, o controlador de teclado i8042 (ou algum de mesma família) é o responsável por receber os scancodes enviados pelo teclado. Esse controlador é gerenciado pelo Kernel Linux segundo o código-fonte descrito em ***drivers/input/serio/i8042.c***.

A correspondência entre teclas e scancodes, entretanto, não é de um para um. Na verdade, um tecla pressionada pode produzir uma sequência de até 6 scancodes[3]. De fato, é tarefa do kernel fazer o *parser* dessa stream de scancodes, interpretando sequências distintas de scancodes como um mesmo **key code**. Essa tarefa está descrita em ***drivers/input/keyboard/atkdb.c*** **.** Dessa maneira, pressionar a tecla 'ESC' no canto superior esquerdo do teclado sempre irá produzir o Key Code 1, independente dos scancodes enviados pelo teclado e a sequência de como tais scancodes foram enviados.

Uma vez definido o key code associado à tecla pressionada, esse é transferido ao sistema de input do Kernel, que irá gerenciar os eventos do teclado segundo especificação descrita em ***drivers/char/keyboard.c*** . Esse módulo é o responsável por viabilizar o uso do padrão Unicode no nível do Kernel, o que é interessante pois permite o uso de diversos layouts de teclado (Key Maps) em tempo de execução do Kernel através do comando **loadkeys**.

Neste ponto, o caractere já está pronto para ser encaminhado para o gerenciador de terminal, conhecido no mundo Unix por **tty**. Essa é a ponte entre o kernel Linux e os processos de usuário, conforme pode ser ilustrado a seguir [4]:



O caractere processado pelo driver do teclado é encaminhado para o emulador de terminal (em substituição aos antigos terminais físicos presentes no início do sistema UNIX). Então, esse caractere é anexado em um ou mais buffers (Line Discipline) do sistema operacional, de maneira que um fluxo de caracteres possa ser editado via alguns comandos básicos rudimentares do Kernel, como *Erase Word, Clear Line*, dentre outros [4]. Através desses buffers, aplicações podem operar ora no modo canônico, ora em modo **RAW** (*Read After Write*), no qual a aplicação é responsável por gerenciar os comandos de edição do fluxo de caracteres. Isso é extremamente comum na maioria das aplicações interativas. Por fim, esses vários buffers ficam à disposição do Driver TTY, responsável pela execução e gerenciamento de processos do usuário em tempo compartilhado.

Abordagens para Keylogging em Linux

Um keylogger, em Linux ou outro sistema UNIX, deve seguir uma de duas abordagens [5] :

- Ser uma aplicação no espaço do usuário, que de alguma maneira intercepta o fluxo de caracteres dentro do Kernel usando permissões de super-usuário;

- Ser um aplicação (ou módulo) dentro do Kernel, que intercepta os key codes em algum ponto do fluxo dos caracteres.

Cada abordagem possui prós e contras. Para os pontos a favor de um keylogging como processo do usuário, destacam-se e independência entre versões do Kernel, a facilidade para acessar a rede e a relativa facilidade em escrever grandes quantidades de dados no sistema de arquivos. Contudo, o programa keylogger deve de alguma maneira conhecer o layout de teclado utilizado, além de ser capaz de isolar interferências no fluxo de caracteres advindos de outros dispositivos que utilizem a mesma porta do hardware (por exemplo, um mouse USB ou PS2).

Em contrapartida, keyloggers de Kernel tendem a ser mais eficientes e fáceis de escrever do que concorrentes em espaço do usuário, uma vez que não são necessárias traduções de key codes para alguma forma legível, tradução de keymaps e outros : há menos trabalho a ser feito. Além disso, tais keyloggers tem

acesso a todas as estruturas do driver TTY, o que permite keylogging sobre todos os usuários loggados na máquina, como processos SSH ou Telnet. Contudo, essa estratégia tende a ser dependente da versão do Kernel Linux para a qual foi concebida, e em particular para o Kernel Linux 2.6, mais difícil de ser implementa uma vez que a tabela de chamadas ao sistema não é mais exportada, de modo que não há maneira fácil de interceptar uma system call. Além disso, tende a ser mais difícil acessar o sistema de arquivos de dentro do Kernel, e inevitavelmente esse terá que ser recompilado para que o keylogger seja operável.

Proposta para Implementação

A proposta desse grupo é implementar um hook keylogger para Linux, que opere como um processo de usuário. Tal abordagem é adotada devido aos fatores de independência de plataformas de Kernel, pela simplicidade em relação ao Kernel em si (recompilar o Kernel pode simplesmente ser inviável em certos ambientes), além da menor complexidade relativa em relação à extensão do código-fonte a ser desenvolvido para futuros upgrades, uma vez que keymaps de teclado, tabelas de keycodes, dentre outras informações, são facilmente acessíveis via internet e incorporáveis ao programa a ser eventualmente extendido. A estratégia, em princípio, é ler diretamente a porta PS2 do teclado com permissão de root, reinterpretando scancodes e obtendo os keycodes gerados, salvando o fluxo de caracteres em algum arquivo de log.

Inicialmente, não está previsto tratamento de erros para outros dispositivos na mesma porta (mouse), sendo que esse aspecto do keylogger – desejável por parte do grupo – está sujeito exclusivamente ao fator tempo demandado na confecção do programa, que deve ser compatível com a data de entrega do projeto segundo a ementa do curso. A implementação será feita em linguagem C, dada a sinergia dessa com o mundo UNIX. Adicionalmente, será estudada a viabilidade de um metakeylogger, capaz ser utilizado tanto para teclados PS2 como USB.

Referências

1. "*Keystroke logging*". Wikipedia, The Free Encyclopedia, 29 de agosto de 2009, 16:11 UTC. Recuperado em 30 de agosto de 2009. http://en.wikipedia.org/wiki/Keylogging
2. "*Linux and the Keyboard*" . Auf Kante, 10 de abril de 2007. Recuperado em 29 de agosto de 2009. http://gunnarwrobel.de/wiki/Linux-and-the-keyboard.html
3. "*The Linux Keyboard Driver*". The Linux Journal, 01 de junho de 1995. Recuperado em 29 de agosto de 2009. http://www.linuxjournal.com/article/1080
4. "*The TTY Demystified*" . Linus Akelson, 25 de julho de 2008. Recuperado em 28 de agosto de 2009. http://www.linusakesson.net/programming/tty/index.php
5. "*Study of Buffer Overflows and Keyloggers in the Linux Operating Systems*". Carrol, Patrick - University of Baltimore, Novembro de 2007.