ESTUDO DE MÓDULOS DE KERNEL DO LINUX PARA KEYLOGGERS

Fernando Leite Barreto[1]; Fabrício Sérgio de Paula[2]
[1]Estudante do Curso de Ciência da Computação da UEMS,

Unidade Universitária de Dourados; E-mail: nando_leite.barreto@hotmail.com
Bolsista da UEMS
[2]Professor do Curso de Ciência da Computação da UEMS;
Unidade Universitária de Dourados; E-mail: fabricio.paula@gmail.com

Área temática: Redes de Computadores e Segurança Computacional

RESUMO

Os *Keyloggers* são aplicativos que monitoram computadores de usuários. No ambiente Linux, o mais utilizado é o LKL (Linux *Keylogger*) que captura as teclas que foram digitadas em seu teclado, armazenando-as em um arquivo no próprio computador que poderá também ser enviado por e-mail. Alguns *keyloggers* têm recursos adicionais como copiar imagens da tela. Muitos utilizam dessa aplicação na realização de delitos e furtos de informações pessoais.

Este projeto propõe realizar um estudo sobre uma maneira de entender melhor como é feita a captura de teclas pelo programa, na tentativa de auxiliar em outras pesquisas no desenvolvimento de módulos de *kernel* que informe quando o computador está sendo monitorado por estes programas, bem como dificultar a leitura desses códigos capturados pelos *keyloggers*.

Para isso, foram realizados profundos estudos a respeito dos seus códigos fontes e sobre a utilização de módulos de *kernel*.

PALAVRA-CHAVE

Segurança. Computação. Redes. Retenção

INTRODUÇÃO

A utilização de *keyloggers*, hoje em dia, é bastante discutida no ramo da informática. Um *keylogger* é uma aplicação que captura e armazena as teclas digitadas por outras aplicações para posterior análise. Devido a isso, ela viola a integridade do usuário, que muitas vezes é alvo de pessoas que a monitora sem que a vítima perceba.

Hoje em dia, grande parte dos antivírus disponíveis detecta a intrusão de *keyloggers*. Na busca de novas formas de detecção pode-se inserir em um módulo, um código que contenha e alerte

sobre a captura de teclas. Caso isso ocorra, deve-se utilizar algum recurso que dificulte a leitura dessas teclas capturadas, como, por exemplo, a criptografia dos dados, que poderá servir de auxilio (Alecrim, 2009).

Esse trabalho tem o escopo de estudar uma maneira de verificar no *kernel* do Linux, versão 2.6 [1], a existência da aplicação de um *keylogger*, utilizando algum procedimento para bloqueá-lo ou dificultar seu entendimento.

MATERIAIS E MÉTODOS OU MÉTODOS

As fontes bibliográficas incluem artigos disponíveis na Internet e o acesso às bibliotecas da UEMS e UFGD. As atividades práticas foram realizadas através do uso de máquinas virtuais e ferramentas de monitoramento como LKL. Foi utilizado o sistema operacional Linux, *kernel* 2.6, e ferramentas/bibliotecas de código aberto e gratuitas, de forma que foram adaptadas de acordo com os requisitos da pesquisa.

RESULTADOS/DISCUSSÃO

O foco do projeto se expande no universo Linux, sistema feito por Linus Torvalds na Universidade de Helsinque, Finlândia, em 1991, com a ajuda de desenvolvedores ao redor do mundo. É um software livre para qualquer alteração no módulo do *kernel* e onde um usuário com um abrangente conhecimento em programação pode, perfeitamente, realizar melhorias neste sistema operacional.

Módulos do *kernel* são partes do mesmo. São carregadas quando solicitadas por algum aplicativo ou dispositivo e descarregadas da memória quando não são mais usadas. A utilização do módulo no *kernel* possui várias funcionalidades, como por exemplo, evitar a construção de um *kernel* com o tamanho muito elevado (estático) que ocupa grande parte da memória com todos os *drivers* compilados e permite que apenas uma parte do *kernel* ocupe a memória somente quando for necessário. (da Silva, 2002).

Os módulos são pedaços de códigos que podem ser carregados e descarregados no *kernel*. Sua utilização é de extrema importância em *device drivers*, extensões aos diversos *frameworks* e subsistemas do *kernel* e também no desenvolvimento de novos modelos de segurança. [2]

[1] Versão do kernel mais recente disponível em www.kernel.org

[2] Exemplos de módulos de kernel disponível em www.4shared.com/file/8HYzCMSC/Exemplos_Modulos.html

Os módulos do *kernel* se encontram no diretório */lib/modules/versão_do_kernel/*.

Os módulos são carregados automaticamente quando o usuário solicita o comando *kmod* no terminal, ou, manualmente, através dos arquivos encontrados no diretório */etc/modules* com os comandos *insmod* ou *modprobe*.

A linguagem utilizada nos módulos é a linguagem C e um erro na programação do módulo pode destruir o sistema de arquivos do usuário e ocasionar um *dump* de memória, ou seja, um *reboot*.

Foi utilizado na instalação de uma máquina virtual o *software Oracle VM Virtualbox* [1] o qual é extremamente útil hoje em dia, pois permite rodar sistemas operacionais dentro de outro sistema, no caso em tela, o sistema operacional virtual utilizado foi o Linux openSUSE 11.2 (i586) com *kernel 2.6.31.5-0.1-default i686*.

Antes de instalar o *kernel* 2.6.36.1 (na época a versão mais atual), foi necessário realizar o download de alguns itens importantes (Goes, 2004). O sistema openSUSE possui um gerenciador de pacotes chamado *YaST*, onde foram realizados todos os *downloads* necessários para a instalação. Os mais recomendados são o *ncurses-dev* (necessário para o *make menuconfig*) ou *qt3-dev* (necessário para o *make xconfig*). Existem diversos *keyloggers* disponíveis na internet. O *Linux Keylogger* (LKL) é um *keylogger* que executa em Linux x86. Ele grava em um arquivo tudo o que passa pela porta do teclado hardware (0x60) e verifica a tecla digitada pelo usuário com o auxilio de keycodes ASCII, um arquivo de mapa de teclado. (Schinaecher, 2010)

A sua última versão é a 0.1.1 e foi lançada em outubro de 2005.[2]

A instalação é, de certa forma, simples. Seus comandos são realizados através do terminal. Depois de fazer o download do arquivo, é necessário descompactá-lo através do comando *tar vzxf lkl-0.1.1.tar.gz.gz.* Após este processo de descompactação, utiliza-se o comando '*cd*' para o diretório contendo o pacote baixado. Caso não destine uma localização específica para o arquivo no momento do *download*, a pasta, provavelmente, estará em *Download*.

Este exemplo cita o modo como se deve entrar no diretório *Download* e na pasta *lkl*: *cd /home/"usuário"/Download/lkl*. Logo em seguida, são necessários os comandos *./configure para* realizar a configuração do pacote para o sistema, *make* para compilar o pacote e *make install* para instalar o pacote compilado.

[1] Para mais informações do *Virtualbox* acesse http://www.virtualbox.org/

[2] Software LKL disponível em http://sourceforge.net/projects/lkl/

Para executar o programa, é necessário entrar como usuário privilegiado (*root*) pelo comando *su* ou *sudo* seguido do nome do programa (comando *./lkl*). Uma senha é solicitada, confirmando a entrada do usuário privilegiado para, desta forma, começar a executar o programa.

Em outros sistemas, como o *Ubuntu* a instalação do LKL é bem mais facilitada. Basta o usuário digitar *sudo apt-get update* para fazer a atualização do sistema, realizar *download* de pacotes essenciais e depois *sudo apt-get install lkl* para baixar e instalar o *keylloger*.

Quando invocado *lkl* sem especificar nenhuma opção, é mostrada uma lista de opções disponíveis:

-- Linux Key Logger vers 0.1.1 --
usage:
-h this help
-l start to log the 0x60 port (keyboard) –b Debug Mode Perhaps it's usefoul
-k <km_file> set a keymap file
-o <o_file> set an output file
-m <email> send logs to <email> every 1k
-t <host> hostname for sendmail server. default is localhost
Example: lkl -l -k keymaps/pt_br -o log.txt

O comando –h exibe uma lista de opções que o programa pode nos oferecer.

O comando –l inicia a gravação do teclado pela porta 0x60, ou seja, pelo teclado físico. Portanto, caso o computador utilizar, por exemplo, um teclado UBS, o programa não captura as teclas digitadas. Para este problema, uma solução seria a utilização de um adaptador que conecta o teclado USB à porta PS2.

O comando *–k <km_file>* especifica qual mapa de teclado *(keymap)* irá ser utilizado. As pastas *keymaps* estão em *lkl/keymaps.* O caminho completo é geralmente */usr/share/LKL/keymaps*. No Linux, existem outros *keymaps* como *fr_km, it_km, us_km, dvrrac_km*, dentre outros.

O comando *–o <o_file>* define o arquivo de saída em que os registros serão gravados. Assim, um exemplo bem típico, poderia ficar assim:

$ sudo lkl-l-k /usr/share/LKL/keymaps/it_km-o /home/user/log.txt

O programa também possui o recurso de ser executado com o console fechado. Basta adicionar o símbolo & no final de linha de comando:

$ sudo lkl-l-k /usr/share/LKL/keymaps/it_km-o /home/user/log.txt&

Para fazer arquivo invisível na qual os registros são mantidos, basta iniciar o seu nome com um ponto (.).

$ sudo lkl-l-k /usr/share/LKL/keymaps/it_km-o /home/user/.log.txt&

O comando *–m <email>* informa o endereço de email para enviar os dados registrados a cada 1k.

O comando *–t <host>* especifica o *host* para o servidor. Por padrão, *localhost*.

O LKL possui cinco códigos essenciais que formam o programa:

No código *lkl.h* são constituídas as bibliotecas utilizadas, as palavras definidas que não poderão ser alteradas, o registro *lkl* que contém as principais variáveis que serão manipuladas pelo programa e a declaração das cinco funções principais utilizadas.

No código *main.c*, também conhecido como a função principal do programa, executa suas primeiras instruções. No começo, verifica se o usuário está em modo *root*, para poder continuar a execução do programa. Após isso, são verificados os argumentos que foram digitados no terminal (*shell*). A função *usage* é executada mostrando as opções do programa disponível e espera que o usuário interaja com o programa podendo habilitar algumas variáveis, dependendo do argumento passado. Quando a variável *lkl.log* é ativada, através do argumento *"l"*, dá acesso ás funções *def_keymap* e *start_log*.

No código *output.c*, está localizada a função *def_keymap* que define o mapa de teclado utilizado pelo *keylloger* na pasta *keymaps* localizada na própria pasta do software. O programa faz a leitura dos arquivos: *"km_file"*, armazenando em *asciitab*, *"km_file"UP*, armazenando em *asciitab_shift* e *"km_file"ALT*, armazenando em *asciitab_alt*.

Primeiramente, antes de executar a *do_output*, o programa executa a função *code2ascii*, que retorna um caractere armazenado no vetor *asciitab*, ou em *asciitab_shift* ou em *asciitab_alt*, dependendo se as teclas *Shift* ou *Alt* estão sendo pressionadas. A função *do_output* é utilizada para gravar no arquivo de saída os caracteres capturados pelo programa. Senão houver um arquivo de saída, o programa imprime no próprio terminal.

No código *lkl.c*, antes da execução da *do_output*, a função *start_log* é executada onde um caracter recebe o valor contido na porta do 0x60 (teclado) e são verificadas algumas condições como se *alt* ou *shift* estivesse pressionada para, então, habilitá-la no registro.

Ocorre que muitos usuários têm muitas dificuldades em realizar a gravação dos caracteres em um arquivo, devido ao *loop* que o programa faz quando a variável *status* recebe a leitura da

porta 0x64 armazenada na variável KEYBOARD_STATUS_PORT e não entra na condição, onde compara se o valor de *status* é igual a 20 (vinte).

Desse modo, foi feita uma nova versão do software, onde é solucionado este problema e está disponível para *download* com comentários sobre o que cada função executa. [1]

E, finalmente, no código *net.c*, a função *snd_mail* é a utilizada para enviar os caracteres coletados via email. A ideia é formar um enorme vetor de caracteres que conterá as teclas capturadas. Neste caso, são usados *sockets*, para conectar-se com o servidor.

[1] Códigos do LKL modificado e comentados em português disponíveis em http://www.4shared.com/file/jjx9yiI6/lkl__modificado_tar.html

CONCLUSÕES

Este trabalho permite melhor compreensão sobre o funcionamento de *keyloggers*, especialmente, o *software* LKL do Linux, através dos comentários nos códigos disponibilizados. A construção de módulos de *kernel* para monitoramento de *keyloggers* é, de certa forma, complicada, devido à ausência de materiais mais abrangentes sobre o assunto. Porém, foram realizados e disponibilizados alguns exemplos para o desenvolvimento de módulos de *kernel* simples, o que pode auxiliar no desenvolvimento de trabalhos futuros.

AGRADECIMENTOS

Agradeço o apoio de minha família, do orientador, do coordenador, dos colegas acadêmicos, da UEMS (Universidade Estadual de Mato Grosso do Sul) pelo auxílio pecuniário, e a todos que, diretamente ou indiretamente, contribuíram para a realização desse trabalho.

REFERÊNCIAS

Goes, Jonas. 2004. Compilando um *Kernel* Linux série 2.6. Disponível em: http://www.dicas-l.com.br/arquivo/compilando_um_kernel_linux_serie_2.6.php

Alecrim, Emerson. 2009. Criptografia. Disponível em: http://www.infowester.com/criptografia.php

Schinaecher, Marcelo. 2010. LKL Linux KeyLogger (pedido do Gaúcho). Disponível em: http://www.dihitt.com.br/n/downloads/2010/05/14/lkl-linux-keylogger-pedido-do-gaucho

da Silva, Gleydson Mazioli, 2002. Guia Foca GNU/Linux. Disponível em: http://www.guiafoca.org/