Posibilidades de infección en Linux

¿SEGURO SEGURO?

Mientras unos dicen que se avecina el gran ataque, otros afirman que no hay nada de qué preocuparse. Pero, ¿cuál es la verdadera historia acerca de la posibilidad de sufrir un virus en sistemas Linux?

POR TOMASZ KOJM

Puede que un sistema Linux no sea tan vulnerable como uno bajo Windows, pero si creemos que los virus para Linux no existen, vamos a tener que recapacitar. Los autores de virus tienen una variada gama de posibilidades de introducir virus en un sistema Linux, aunque el daño puede limitarse si somos mínimamente cuidadosos y seguimos algunas sencillas reglas. En este artículo vamos a describir algunos ejemplos de cómo funcionan los virus para Linux, y daremos algunos consejos para mantener nuestros equipos seguros.

Un Teórico Virus para Linux

La mayoría de las distribuciones incluyen *gzexe*, un pequeño programa que comprime archivos ejecutables y los descomprime automáticamente al ejecutarlos. Por ejemplo, podemos copiar */bin/date* en */tmp* y ejecutar *gzexe /tmp/date* para comprimir el archivo ejecutable. El tamaño de */bin/date* y de */tmp/date* debería ser distinto, y además, el de este último considerablemente menor. Ejecutamos ambos archivos. ¿Notamos alguna diferencia?

Todo ejecutable comprimido con gzexe incluye una pequeña sección de código al principio del archivo. Si abrimos */tmp/date* con nuestro editor favorito, veremos que no es otra cosa sino un simple script de shell. Bajo esta sección de código se sitúan la información en binario con el ejecutable comprimido. El código shell es el responsable de descomprimir la información en un archivo temporal y de ejecutarlo. Todo este proceso es transparente para el usuario, y con los veloces ordenadores de hoy día, el retardo en ejecutar archivos comprimidos es marginal.

Imaginemos ahora una sección de código modificada que haga lo siguiente antes de descomprimir y ejecutar el código original:

- busca aleatoriamente en $PATH algún archivo ejecutable que se pueda escribir (o un archivo que sea propiedad del usuario en uso), que no sea un script de shell.
- comprime el archivo ejecutable (puede incluir el código del script de shell de gzexe) e inserta la cabecera modificada en su interior.

Esta sección de código se ajusta a la definición de virus que aparece en la Wikipedia: "un programa de ordenador que puede infectar otros programas modificándolos para incluir una copia de sí mismo" [1]. Llamémosle Linux.Gzipper.

Como muestra este sencillo ejemplo, no es un gran desafío el escribir un pequeño virus para Linux. No debería sorprendernos que los autores de virus puedan usar métodos más sofisticados. El formato ELF (Executable and Linking) de los archivos ejecutables es muy similar al formato PE (Portable Executable) que usa Microsoft Windows y proporciona casi la

TEMA DE PORTADA

Listado 1: Comenzamos con un Archivo Infectado

```
01 testuser@testsystem:~/testfiles$ ls -l
02 total 727
03 -rwxr-xr-x 1 testuser testuser 49084 Sep 4 03:32 cp
04 -rw-r--r-- 1 testuser testuser 651 Jul 28 2004 crontab
05 -rwxr-xr-x 1 testuser testuser 88038 Nov 6 17:51 date.infected
06 -rw-r--r-- 1 testuser testuser 1489 Feb 10 2004 fam.conf
07 -rw-r--r-- 1 testuser testuser 292 Jun 18 02:05 hosts
08 -rwxr-xr-x 1 testuser testuser 71996 Sep 4 03:32 ls
09 -rw-r--r-- 1 testuser testuser 1426 Nov 6 01:44 passwd
10 testuser@testsystem:~/testfiles$ clamscan --no-summary
11 /home/testuser/testfiles/cp: OK
12 /home/testuser/testfiles/ls: OK
13 /home/testuser/testfiles/crontab: OK
14 /home/testuser/testfiles/hosts: OK
15 /home/testuser/testfiles/fam.conf: OK
16 /home/testuser/testfiles/date.infected: Linux.Rst.A FOUND
17 /home/testuser/testfiles/passwd: OK
```

misma funcionalidad. Es decir, los autores de virus podrían usar muchas técnicas avanzadas de infección de archivos ejecutables que han desarrollado para atacar sistemas Windows durante la última década. Por supuesto, ya existen un buen número de virus que infectan el sistema ELF. Es más, algunos virus pueden incluso infectar tanto archivos PE como ELF. Sin embargo, incluso los virus para Linux más avanzados se enfrentan al mismo problema que se encontró nuestro pequeño Gzipper: no es tan sencillo dañar un sistema Linux.

Protección Nativa Anti-Virus

Los autores de virus explotan el hecho de que la mayoría de los usuarios de sistemas operativos comerciales están acostumbrados a trabajar con un alto nivel de privilegios que permite la manipulación directa de recursos críticos del sistema.

En el caso de Linux, y de UNIX en general, es un principio fundamental que el usuario sólo debería usar la cuenta de root para acciones de administración y nunca en el uso cotidiano. Siempre que se obedezca esta regla, la mayoría de los virus no podrán hacer un daño global, ya que el mecanismo de permisos de archivos protege al sistema y a los ficheros de los usuarios. Por supuesto, un virus podría tratar de realizar una escalada de privilegios, pero en este caso, los sistemas con agujeros ya están de todos modos a merced de otros tipos de ataque, no solamente de virus.

Nuestro teórico Gzipper busca en *$PATH* archivos ejecutables que pueda infectar. Cuando Gzipper se ejecuta por un usuario sin privilegios, sólo será capaz de infectar archivos ejecutables que pertenezcan o se puedan escribir por este usuario, y en la mayoría de los casos no será capaz de extenderse más allá del entorno de dicho usuario. Desafortunadamente, existen algunas distribuciones de Linux "de fácil uso" que promocionan el modelo erróneo de un único usuario con altos privilegios, lo que abre la puerta a ataques de virus y permite a éstos infectar partes vitales del sistema operativo. Estas distribuciones podrían llamarse "de fácil infección" en lugar de "de fácil uso".

Las Diversas Caras de Linux

Otra cosa que dificulta la propagación a los virus es la diversidad de distribuciones de Linux y las arquitecturas soportadas, así como las numerosas diferencias técnicas entre ellas. Por supuesto, un virus compilado para la arquitectura x86 no funcionará en un SPARC y viceversa. Incluso los virus "portables" escritos en lenguajes de script como Perl o la shell pueden fracasar en su intento por ejecutarse en función de los elementos que no estén disponibles en el sistema de la víctima.

Propagar los problemas

Los virus, al contrario que los gusanos, no tienen mecanismos para replicarse entre sistemas. Sólo pueden propagarse junto al archivo anfitrión, contaminando otros archivos en el proceso. Hoy día, la mayoría de los usuarios y administradores de sistemas Linux instalan software de sus distribuciones o de fuentes oficiales que se consideran fiables. Además de esto, los paquetes oficiales suelen estar firmados

Listado 2: Propagamos la Infección

```
01 testuser@testsystem:~/testfiles$ ./date.infected
02 Sun Nov 6 18:02:46 CET 2005
03 testuser@testsystem:~/testfiles$ ls -l
04 total 1010
05 -rwxr-xr-x 1 testuser testuser 97890 Nov 6 18:02 cp
06 -rw-r--r-- 1 testuser testuser 651 Jul 28 2004 crontab
07 -rwxr-xr-x 1 testuser testuser 88038 Nov 6 17:51 date.infected
08 -rw-r--r-- 1 testuser testuser 1489 Feb 10 2004 fam.conf
09 -rw-r--r-- 1 testuser testuser 292 Jun 18 02:05 hosts
10 -rwxr-xr-x 1 testuser testuser 120802 Nov 6 18:02 ls
11 -rw-r--r-- 1 testuser testuser 1426 Nov 6 01:44 passwd
12 testuser@testsystem:~/testfiles$ clamscan --no-summary
13 /home/testuser/testfiles/cp: Linux.Rst.A FOUND
14 /home/testuser/testfiles/ls: Linux.Rst.A FOUND
15 /home/testuser/testfiles/crontab: OK
16 /home/testuser/testfiles/hosts: OK
17 /home/testuser/testfiles/fam.conf: OK
18 /home/testuser/testfiles/date.infected: Linux.Rst.A FOUND
19 /home/testuser/testfiles/passwd: OK
```

Listado 3: Hemos Creado una Puerta Trasera

```
01 testuser@testsystem:~/testfiles$ ps aux
02 USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
03 [...]
04 testuser 28000 0.0 0.1 2116 876 ? S 18:02 0:00 ./date.infected
05 [...]
06 testuser@testsystem:~/testfiles$ netstat -upan
07 [...]
08 udp 0 0 0.0.0.0:5503 0.0.0.0:*28000/date.infected
```

digitalmente y se pueden verificar antes de la instalación.

Desafortunadamente, incluso el mejor mecanismo de prevención no puede evitar un error humano.

En septiembre de 2005, la versión oficial coreana de Mozilla Suite 1.7.6 y Thunderbird 1.0.2 para Linux resultó infectada con el virus Linux.RST.B. El incidente fue bastante serio porque el navegador Web generalmente se instala globalmente desde la cuenta de root para hacerlo disponible a todos los usuarios del sistema. La ejecución de la suite infectada desde una cuenta con privilegios podría permitir al virus infectar fácilmente archivos del sistema. La Fundación Mozilla publicó un aviso de seguridad, recomendando a los usuarios coreanos que hubiesen instalado productos infectados que escaneasen sus sistemas con un antivirus [2].

Estos incidentes de momento son bastante raros, pero es probable que se intensifiquen en el futuro a medida que se incremente el software puesto a disposición por parte de terceras fuentes.

El Auténtico Problema

Los virus de ordenador a menudo llevan un *payload*, que es la acción especial que desarrollan después de propagarse. En nuestro ejemplo de Linux.Gzipper era comprimir los archivos objetivo antes de infectarlos. Aunque, en algunos casos, incluso este tipo de acciones tan inocentes pueden causar un serio malfuncionamiento del sistema, la mayoría de los virus en el mundo real no son tan simpáticos como Gzipper.

La primera versión del virus RST se descubrió a finales de 2001. El nombre proviene de "Remote Shell Trojan". RST trata de infectar archivos ejecutables en el directorio actual, y si tiene suficientes privilegios, también infecta los archivos de sistema de */bin*.

El Listado 1 muestra un directorio con unos cuantos archivos limpios y uno infectado con el virus. Si ejecutamos *date.infected* como usuario sin privilegios, provocamos la infección del directorio actual (Listado 2). Además, el virus activa su carga y arranca un servidor en segundo plano que se pone a la escucha del socket UDP (Listado 3). En este momento, un atacante con conocimientos de un protocolo especial puede tomar control por la puerta trasera y arrancar un shell remoto en el sistema infectado.

Listado 4: Compilamos Hello, World

```
01 testuser@testsystem:~/hello$ cat hello.c
02 #include <stdio.h>
03
04 int main(int argc, char **argv)
05 {
06 printf("Hello world!\n");
07 return 0;
08 }
09 testuser@testsystem:~/hello$ gcc hello.c -o hello1
10 testuser@testsystem:~/hello$ ./hello1
11 Hello world!
12 testuser@testsystem:~/hello$ ls -l
13 total 12
14 -rw-r--r-- 1 testuser testuser 100 Nov 6 18:46 hello.c
15 -rwxr-xr-x 1 testuser testuser 7340 Nov 6 18:50 hello1
```

Algunos de los virus Linux que existen implementan técnicas de infección muy interesantes. Linux.Svat es un ejemplo de este último tipo. En lugar de infectar directamente, intenta modificar el sistema operativo para crear archivos infectados. El Listado 4 muestra el proceso de compilación de un programa "Hello, World" estándar. La idea de Svat se basa en el diseño del compilador. Cuando GCC se encuentra una macro *#include <file.h>*, busca en primer lugar el archivo cabecera *file.h* en */usr/local/include* y más tarde en */usr/include*, que es el directorio donde están instalados todos los archivos cabecera importantes. El archivo *stdio.h* es uno de los archivos cabecera más usados.

Cuando el archivo infectado por Linux.Svat se ejecuta desde la cuenta de root, se instala una nueva cabecera en */usr/local/include*. El nuevo *stdio.h* incluye el original y además redefine la función del sistema *close()*, que a partir de ese momento llama a la rutina del virus *virfunc()* antes de cerrar los descriptores del archivo. La rutina tiene bugs y causa una violación de segmento si no tiene permiso de escritura en el directorio */usr/local/include*. Este desliz en el código limita las oportunidades del virus para replicarse. La rutina de infección se incluirá en cada nuevo archivo que se compile haciendo uso de *stdio.h*. Debido a que nuestro archivo de ejemplo *hello.c* no llama a la función close(), el código del virus que tenemos en *hello2* no se activará nunca.

Las Reglas de Oro

Las reglas para protegernos de virus Linux son similares a las reglas para otros sistemas:

1.No usar nunca la cuenta de root para el trabajo rutinario.

2.Evitar la ejecución de archivos binarios de origen desconocido. Verificarlos previamente con escáners de rootkits y virus.

3.Verificar cuidadosamente cada archivo antes de ejecutarlo desde una cuenta de root.

4.Mantener el sistema operativo actualizado. Instalar regularmente las actualizaciones de seguridad.

5.Asegurar nuestro entorno con contraseñas difíciles de quebrantar y otras protecciones.

6.Comprobar los cambios en el sistema con herramientas de integridad. ■