

SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática – Complementos de Ing. del Software
Curso 2018-19

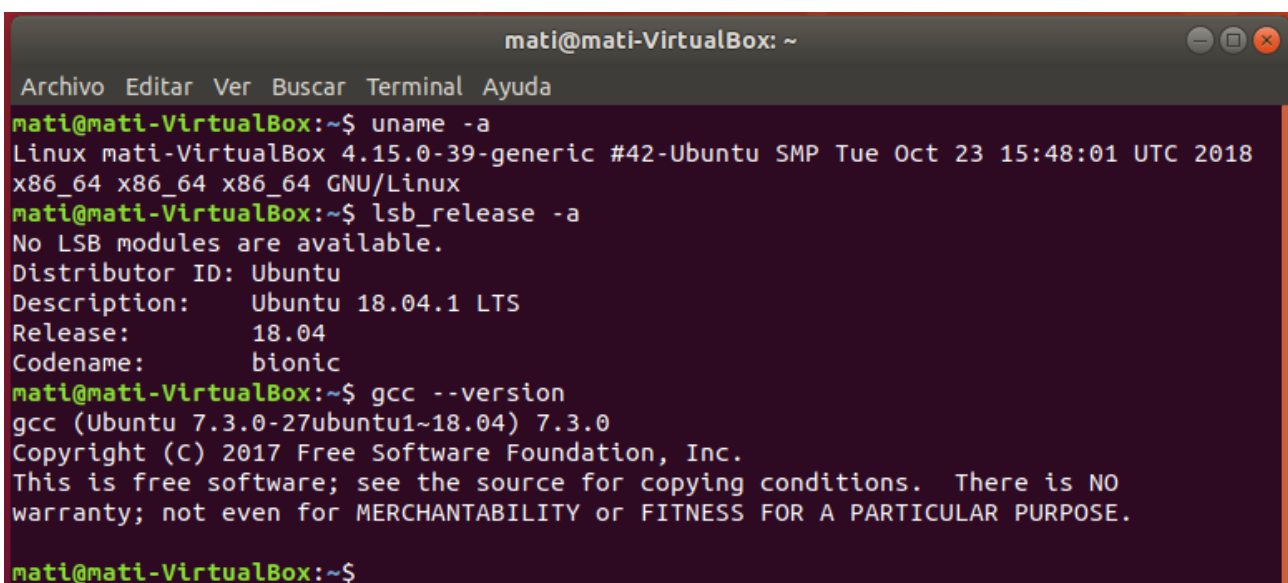
Práctica [2]. Ingeniería inversa en Linux

Sesión [2]. Explotaciones y protecciones del formato ELF

Autor¹: Matilde Cabrera González

Ejercicio 1.

Para el sistema que utilizas, indica la arquitectura, distribución y compilador que utilizas e indica que protecciones se utilizan de cara a proteger un binario ELF.

A screenshot of a terminal window titled 'mati@mati-VirtualBox: ~'. The terminal shows the output of several commands: 'uname -a' returns 'Linux mati-VirtualBox 4.15.0-39-generic #42-Ubuntu SMP Tue Oct 23 15:48:01 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux'; 'lsb_release -a' returns 'No LSB modules are available. Distributor ID: Ubuntu Description: Ubuntu 18.04.1 LTS Release: 18.04 Codename: bionic'; and 'gcc --version' returns 'gcc (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0 Copyright (C) 2017 Free Software Foundation, Inc. This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.'

```
mati@mati-VirtualBox: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
mati@mati-VirtualBox:~$ uname -a  
Linux mati-VirtualBox 4.15.0-39-generic #42-Ubuntu SMP Tue Oct 23 15:48:01 UTC 2018  
x86_64 x86_64 x86_64 GNU/Linux  
mati@mati-VirtualBox:~$ lsb_release -a  
No LSB modules are available.  
Distributor ID: Ubuntu  
Description:    Ubuntu 18.04.1 LTS  
Release:        18.04  
Codename:       bionic  
mati@mati-VirtualBox:~$ gcc --version  
gcc (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0  
Copyright (C) 2017 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
mati@mati-VirtualBox:~$
```

Para proteger el binario ELF podemos:

Usar herramientas de análisis como ELF parsing toolkit, binary análisis toolkit...

El sistema en si tiene:

Buffer overflow

Stack not executable

Battery protection bypass

Address Space Randomization

ASLR check

¹ Como autor declaro que los contenidos del presente documento son originales y elaborados por mi. De no cumplir con este compromiso, soy consciente de que, de acuerdo con la “Normativa de evaluación y de calificaciones de los estudiantes de la Universidad de Granada” esto “conllevará la calificación numérica de cero ... independientemente del resto de calificaciones que el estudiante hubiera obtenido ...”

Ejercicio 2.

Podemos mejorar el siguiente virus:

a) Escribiendo un mejor escaner/limpiador para él.

Uso dd, copia archivos byte a byte y podemos especificar el offset de skip que en este caso sería el número de bytes VIRUS_SIZE que figura en el código de lx3k2.

Bs depende del tamaño del archivo

```
dd if=$INFECTED.vx of=$INFECTED count=$ORIG_SIZE skip=$VIRUS_SIZE bs=1
```

b) Añadir `#ifdef` para comprobar la arquitectura de forma que cambie de forma automática el valor

EM_386 en tiempo de compilación para adaptarlo al sistema donde estemos.

Ejecutamos “locate elf.h”, encontramos nuestro archivo a modificar en la ruta” /usr/include/elf.h”, incluimos en el mismo el código necesario:

```
#if _GNUC_
    #if _x86_64_||_ppc64
        #define 64ENV
    #else
        #define 32ENV
    #endif
#endif
```