CCOM4086 sec 0U1 - enero 2020
Laboratorio 5: Desactivar la Bomba Binaria
Fecha de Entrega: 13 de abril 2020

1 Introduction

En este laboratorio vas a poner en práctica tu conocimiento en el lenguaje de ensamblador para desactivar una bomba binaria. Una bomba binaria es un programa que consiste de una secuencia de fases. En cada fase se espera que entres una cadena de caracteres (string) particular en el `stdin`. Si entras la cadena correcta *desactivas* esa fase y puedes pasar a la siguiente. Pero si entras la incorrecta la bomba *explota*, el programa imprime `"BOOM!!!"` y acaba. La bomba se desactiva cuando todas las fases han sido desactivada. Tu misión es desactivar la bomba que te toque (hay una diferente para cada estudiante).

2 Recoge tu Bomba

1. Busca tu bomba apuntando tu browser a:

   `http://ada.uprrp.edu:8080/`

2. Llena la forma para pedir tu bomba. Entra tu nombre de usuario y tu email y dale al botón de Pídela. El servidor construirá tu bomba y generará un archivo `tar` llamado `bomb`k`.tar`, donde k es un número que identifica tu bomba de forma única.

3. Graba el archivo `bomb`k`.tar`

4. Envia el archivo `tar` a ada usando `sftp`.

5. Ejecuta el comando `tar -xvf bombk.tar`. Esto creará el directorio `./bomb`k con los siguientes archivos:

   - `README`: Identifica la bomba y su dueñ@.
   - `bomb`: El ejecutable de la bomba binaria.
   - `bomb.c`: El código fuente de la rutina main de la bomba.

3 Desactiva tu Bomba

Tu misión es desactivar tu bomba.

Tienes que hacer el laboratorio en ada (el ejecutable verifica que estés en ada).

Puedes usar todas las herramientas que se te ocurran. La mejor forma es usar tu debugger favorito para ir paso a paso sobre el binario desensamblado. Asegúrate leer la sección de **ayudas**.

Cada vez que te explote la bomba pierdes 0.5 puntos hasta un máximo de 20 puntos. Así que hay consecuencias de explotar la bomba.

Las primeras cuatro fases valen 10 puntos cada una, las Fases 5 y 6 valen 15 puntos cada una para un total de 70 puntos.

Aunque las fases son cada vez más difíciles, lo que vas aprendiendo te permitirá superar estas dificultades. Sin embargo, la última fase es súper retante así que no esperes hasta el último momento para comenzar tu misión.

Al correr tu bomba tienes la opción de proveer un archivo con las soluciones. Es decir si arrancas el programa con el comando:

```
linux> ./bomb psol.txt
```

el programa va a leer las lineas en el archivo `psol.txt` (una linea por cada fase) hasta llegar a EOF (end of file) y entonces vuelve a estar listo para recibir input de `stdin`. Esto te permite almacenar las contestaciones de las fases que ya desactivastes y no tener que escribirlas cada vez que arranques la bomba. Si usas `gdb`, ya dentro del ambiente puedes correr el programa con el archivo de la siguiente manera:

```
gdb> run psol.txt
```

Para evitar accidentes (explosiones de la bomba) tienes que aprender como detener la ejecución del programa e ir paso a paso en el código desensamblado usando breakpoints en el debugger. Además, tienes que aprender como inspeccionar los registros, la memoria y los estados de la memoria. Aprender a usar un debugger es fundamental para tu carrera (me lo agradecerás luego).

4 Entrega

No hay una entrega explicita. La bomba notifica automáticamente como vas progresando mientras trabajas. Puedes ver el progreso en la página:

```
http://ada.uprrp.edu:8080/scoreboard
```

5 Ayudas

Hay muchas formas de desactivar tu bomba. Puedes examinarla en detalle sin correr nunca el programa y descifrar exactamente lo que hace. Aunque muy útil no es muy fácil de hacer. La mejor

y más rápida forma de desactivarla es usar el debugger e ir paso a paso en el código desensamblado usando la información que vayas obteniendo.

Hay muchas herramientas diseñadas para ayudarte a entender como trabaja un programa y además para descifrar que está mal cuando algo no funciona. Aquí se incluye una lista de algunas de estas herramientas.

- `gdb`

  El debugger de GNU es un debugger que se usa desde el terminal y está disponible en virtualmente cualquier plataforma. Te permite ir linea por linea, examinar registros y memoria, mirar a la vez el código fuente y el ensamblador, poner breakpoints, puntos de observación de la memoria (watch points) y escribir scripts. Existen varios programas que son interfaces gráficas de `gdb` como `ddd` que pueden hacerte la vida mucho más fácil.

  En la página de moodle de la clase hay un enlace a un documento que resume los principales comandos de `gdb`.

- `objdump -t`

  Este comando imprime la tabla de símbolos de tu programa. La tabla de símbolos incluye el nombre de todas las funciones y variables globales del programa, además de todas las funciones que el programa llama con sus direcciones. Estos nombres te pueden dar algunas pistas.

- `objdump -d`

  Este comando desensambla el programa y lo presenta en pantalla. Aquí puedes ver los detalles de cada función del programa. Leer el lenguaje de ensamblador te dice como funciona la bomba.

  Aunque `objdump -d` te da mucha información no te da toda la historia. Las llamadas a funciones del sistema se presentan de forma críptica. Por ejemplo, una llamada a la función `sscanf` aparece como:

  ```
  8048c36:  e8 99 fc ff ff  call   80488d4 <_init+0x1a0>
  ```

  Para saber que la llamada fue a `sscanf` necesitas desensamblar en `gdb`.

- `strings`

  Este comando te muestra todas las cadenas de caracteres imprimibles de la bomba.

Recuerda que linux tiene los comandos `apropos`, `man` e `info` que son muy útiles. Además, la web tiene infinidad de información y recursos relacionados con estas herramientas. Por último, si estás trancad@ pregúntale al profesor. Este mensaje ...