CI-0118 Lenguaje Ensamblador, g01

Tarea programada #1

I/2021

"You can bomb the world to pieces,
but you can't bomb it into peace. Power to the peaceful."
— Michael Franti

Motivación

Un ente malvado plantó una serie de "bombas binarias" en el sitio de Mediación Virtual con el objetivo de poner en peligro sus notas del curso. La razón: nunca la sabremos.

Una bomba binaria es un programa compuesto por una serie de fases o etapas. Todas las etapas deberán ser ejecutadas correctamente para desactivar la bomba. Cada una de las fases espera una cadena de texto en particular de la entrada estándar `stdin`. Al ingresar una cadena correcta la fase estará superada y el dispositivo de activación de la bomba pasará a la siguiente fase. Cada fase superada lo acercará a *desactivar* la bomba por completo. De lo contrario (es decir, no introduce la cadena de texto correcta), la bomba explotará imprimiendo "BOOM!" y después terminará su ejecución.

Hay demasiadas bombas. Por lo que hemos decidido darle a cada estudiante una bomba para desactivar. Su misión, que no tiene otra opción más que aceptarla, es desactivar la bomba que le corresponde antes de la fecha límite. A continuación, se describirá el proceso para desactivar las bombas. ¡Buena suerte!

Paso 1

Se puede obtener accediendo en un navegador la dirección: http://18.219.39.3:15213

Esto le mostrará una página para hacer la solicitud de su bomba. Se recomiendo usar Chrome. Ingrese su número de carnet de la U y su correo electrónico. Luego presione el botón de "Submit". El servidor le construirá una bomba personalizada y le devolverá un archivo tar llamado `bombk.tar`, donde k es el identificador único de cada bomba. Por tanto, el mecanismo de desactivación de cada bomba es diferente.

Guarde el archivo `bombk.tar` en un directorio protegido donde planea trabajar. Desde una terminal ejecute el comando `tar –xvf bombk.tar`. Esto le creará un directorio llamado `./bombk` conteniendo los siguientes archivos:

- README: Identifica la bomba y a su usuario.
- bomb: El ejecutable de la bomba binaria.
- bomb.c: Código fuente con la rutina principal de la bomba y un saludo del ente malvado que plantó la bomba.

Paso 2

Su misión, como ya lo sabe, es desactivar la bomba.

Se recomienda hacer su tarea en una máquina con sistema operativo Linux, preferiblemente Ubuntu. Se pueden usar múltiples herramientas para desactivar estas bombas. Por favor, vea la sección de pistas para algunas ideas de por dónde comenzar. La mejor manera de resolver esta tarea es usando su *debugger* favorito. Cada vez que su bomba explote (porque introdujo una cadena incorrecta) se enviará una notificación al servidor, y perderá 0.5 puntos (para un máximo de 20 puntos perdidos por explotar las bombas) en el resultado final de la tarea programada. Así que deben evitar que la bomba explote para no perder puntos. Cada vez que la bomba explota se vuelve a armar para que intente de nuevo. Esto sucederá hasta que haya resuelto todas las fases, o bien haya perdido el curso (¡mentira!, el curso no se pierde con un cero en esta tarea).

Las primeras cuatro fases tienen un valor de 10 puntos cada una. Las fases 5 y 6 son un poco mas difíciles, entonces valen 15 puntos cada una. La puntuación total es de 70 puntos.

Aunque las fases pueden aumentar progresivamente su dificultad, usted aumenta sus capacidades y conocimientos cada vez que avanza en la tarea. Sin embargo, la ultima fase la más difícil, así que no se espere al último minuto para comenzar a trabajar en esta tarea.

La bomba ignora líneas vacías de entrada. Si usted corre la bomba con el siguiente comando, por ejemplo:

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

Entonces la bomba considerará las líneas de psol.txt hasta alcanzar el EOF (end of file), y después seguirá leyendo de la entrada estándar. En algunos momentos de debilidad, el ente malvado agregó esta posibilidad para que no tengan que escribir de nuevo las respuestas a cada una de las fases que ya se resolvieron.

Para evitar detonar la bomba accidentalmente, necesitará aprender como avanzar instrucción por instrucción a través del código ensamblado (utilizando *breakpoints* en el depurador). También, necesitará aprender a inspeccionar el contenido de los registros y el estado de la memoria. Uno de los efectos colaterales de hacer esta tarea es que usted aprenderá a usar muy bien el *debugger*. Esta es una habilidad esencial que le ayudará a obtener un buen salario al graduarse (o eso esperamos).

Entrega

Esta tarea es individual. Todos los materiales son electrónicos. Cualquier duda o corrección será anunciada en Mediación Virtual (extraoficialmente también el chat de WhatsApp del grupo).

No existe una entrega explícita (es decir, no debe subir ningún código a Mediación Virtual o entregar documento alguno). Dado que la bomba notificará automáticamente sobre el progreso de su tarea mientras trabaja en ella, así podremos ponerle una calificación. De esta forma, 70 puntos equivalen a una nota de 100 en la tarea. Puede mantener el rastro de su trabajo viendo el tablero de resultados en http://18.219.39.3:15213/scoreboard

Esta página se actualiza continuamente conforme trabajan en la tarea.

Se considerará la tabla de puntuación del martes 18 de septiembre a las 10.00 am para asignar la calificación de esta tarea. En el primer examen parcial se evaluarán las etapas 1, 2 y 3 de esta tarea.

Pistas (¡Leálas por favor!)

Existen múltiples maneras de desactivar o desarmar la bomba. Puede analizarla con gran detalle con ni siquiera correr el programa, y darse cuenta exactamente cómo funciona. Es una técnica útil, pero no siempre es la manera mas apropiada. También, se puede usar algún *debugger*, en el que se puede ejecutar el código paso a paso, y usar esta información para desactivarla. Quizá, esta última sea la manera más rápida de hacerlo.

Hacemos la petición de que por favor no haga ataques de fuerza bruta. Usted podría hacer un programa que intente cada posible hilera hasta encontrar la adecuada para cada fase. Pero esto no es muy buena idea por múltiples razones:

- Pierde 0.5 puntos (hasta un máximo de 20 puntos) cada vez que su hilera es incorrecta y la bomba explota.
- Cada vez que responde incorrectamente, se envía un mensaje al servidor. Esto podría saturarlo rápidamente, y causar que el servidor le quite el acceso a su computadora.
- No hemos dicho que tan largas son las hileras, tampoco hemos dicho si tienen caracteres especiales o números. Aunque usted suponga que son menos de 80 caracteres y que únicamente contiene letras, entonces usted tendría 26^{80} posibilidades para cada fase. Esto tomaría mucho tiempo y no terminaría para cuando tiene que terminar la tarea.

Existen múltiples herramientas que están diseñadas para entender como funcionan los programas, y que está mal en ellos para que no funcione. Aquí hay una lista de herramientas que usted puede encontrar útiles para analizar la bomba, y pistas en como usarlas.

- gdb
  El *GNU debbuger* es una herramienta de línea de comandos disponible en prácticamente cualquier plataforma virtual. Usted puede rastrear un programa línea por línea, examinar la memoria y los registros, ver tanto el código fuente como el código ensamblador (nosotros no estamos dando los códigos fuentes completos), poner *breakpoints*, ver puntos especíıficos de la memoria y hasta escribir scripts. Considere que:

  – Para evitar que la bomba explote cada vez que usted escribe una hilera incorrecta, le gustará aprender a poner *breakpoints*.
  – Para documentación en línea, escriba “help” dentro de gdb, o escriba “man gdb” o “info gdb” en una línea de comandos de Linux.

- objdump -t

  Esto imprimirá la tabla de símbolos de la bomba. Esta incluye el nombre de las funciones y de las variables globales, e inclusive las direcciones de memoria de estas funciones. Quizá aprenda algo viendo el nombre de las funciones.

- objdump -d

  Este sirve para desensamblar el código de la bomba. Leer el código en ensamblador le puede ayudar a entender como funciona la bomba.

  Aunque objdump -d da mucha información, este no cuenta toda la historia. Los llamados a funciones del sistema son desplegados de manera críptica. Por ejemplo, un llamado a sscanf puede verse como:

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

  Para determinar que la llamada fue a la función sscanf, necesitara utilizar gdb inevitablemente.

- strings
  Esto mostrará todas las cadenas imprimibles dentro de la bomba.

¿Buscando alguna herramienta particular? ¿Documentación? No olvide que los comandos man e info son sus amigos. En particular, man ascii puede ser de gran ayuda. info gas le mostrará más información sobre el ensamblador de GNU. Si se siente pegado, siéntase libre de pedir ayuda al profesor.

Esta tarea esta basada en los laboratorios propuestos en el libro *Computer Systems: A Programmer’s Perspective.*