66:20 Organización de Computadoras
Trabajo práctico 1: Conjunto de instrucciones MIPS

1. Objetivos

Familiarizarse con la programación en assembly y el concepto de ABI, explotando vulnerabilidades del stack y la función `gets()`.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 8), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo[1], y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Recursos

Usaremos el programa GXemul[1] para simular el entorno de desarrollo: una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD[2]. Además se utilizarán las herramientas `gcc` [5], `objdump` [4], y `gdb` [3].

5. Descripción

5.1. Gera's Insecure Programming

Gerardo Richarte escribió hace ya tiempo unos ejercicios destinados a fomentar la comprensión del funcionamiento de los programas compilados en C y sus

[1] `http://groups.yahoo.com/group/orga6620`

problemas de seguridad[2]. Eventualmente, muchos sistemas operativos agregaron protecciones que impiden los ataques sugeridos en los ejercicios, pero éstos siguen siendo una excelente introducción a las vulnerabilidades de software. Si bien un estudio completo de todo lo que puede fallar en un programa en C está fuera del alcance de este trabajo y de la materia, los problemas más introductorios se ocupan del stack y resultan apropiados para la comprensión del funcionamiento del éste y sus problemas asociados.

El trabajo consiste en conseguir que los programas `stack1.c` a `stack5.c` impriman la frase "you win!", cuando normalmente no lo harían, aprovechando que utilizan la función `gets` para llenar un buffer de largo fijo y conocido. Contamos con la posibilidad de correr el programa bajo `gdb` para examinar la ubicación del programa en memoria, así como los valores de los registros y el contenido del stack, teniendo en cuenta que la solución debe funcionar ejecutando el programa por fuera de `gdb`. No usar los programas que están en el mirror: los originales fueron modificados para que se puedan explotar en MIPS, y en el caso de stack4.c hay modificaciones gratuitas.

6. Programas a subvertir

6.1. stack1.c

```
/* stack1-stdin.c                                *
 * specially crafted to feed your brain by gera */

#include <stdio.h>

int main() {
        char buf[80];
        int cookie=1;

        printf("buf: %08x cookie: %08x\n", &buf, &cookie);
        gets(buf);
        printf ("cookie: %08x\n", cookie);

        if (cookie == 0x41424344)
                printf("you win!\n");
}
```

6.2. stack2.c

```
/* stack2-stdin.c                                *
 * specially crafted to feed your brain by gera */

#include <stdio.h>

int main() {
        char buf[80];
        int cookie;

        printf("buf: %08x cookie: %08x\n", &buf, &cookie);
```

[2]El link original (en Core Security) ya no responde, pero hay al menos un mirror[6].

```
        gets(buf);
        printf("cookie: %08x\n", cookie);
        if (cookie == 0x01020305)
                printf("you win!\n");
}
```

6.3. stack3.c

```
/* stack3-stdin.c                                *
 * specially crafted to feed your brain by gera */

#include <stdio.h>

int main() {
        char buf[80];
        int cookie;

        printf("buf: %08x cookie: %08x\n", &buf, &cookie);
        gets(buf);
        printf("cookie: %08x\n", cookie);
        if (cookie == 0x01020005)
                printf("you win!\n");
}
```

6.4. stack4.c

```
/* stack4-stdin.c                                *
 * specially crafted to feed your brain by gera */

#include <stdio.h>

void stack4();

int main() {
        stack4();
        return 0;
        }

void stack4(){

        char buf[80];
        int cookie;

        printf("buf: %08x cookie: %08x\n", &buf, &cookie);
        gets(buf);
        printf("Cookie: %08x\n",cookie);
        if (cookie == 0x000d0a00)
                printf("you win!\n");
}
```

6.5. stack5.c

```
/* stack5-stdin.c                                *
 * specially crafted to feed your brain by gera */
```

```
#include <stdio.h>

void stack5(){

        char buf[80];
        int cookie;

        printf("buf: %08x cookie: %08x\n", &buf, &cookie);
        gets(buf);
        printf("Cookie: %08x\n",cookie);
        if (cookie == 0x000d0a00)
                printf("you lose!\n");
}
```

7. Tareas a realizar

Para cada uno de los programas se debe realizar la siguiente tarea:

7.1. Análisis del stack

Hacer un diagrama del stack con la dirección y tamaño de cada elemento. Determinar cómo debería ser el stack luego del ingreso de texto por `gets`. Justificar.

7.2. Diseño de la entrada a proporcionar al programa

Determinar cómo debe ser el string de entrada al programa, y cómo ingresarlo. Justificar. Crear la entrada correspondiente, ya sea con un editor de texto/hexadecimal o un programa escrito ad hoc.

7.3. Corridas de prueba

Verificar que la entrada generada causa que el programa en cuestión imprima "you win!".

8. Informe

El informe deberá incluir:

- Diagramas indicando el stack de cada programa antes y después del ingreso del string.
- Documentación relevante al diseño y generación de cada entrada;
- Un archivo por programa con la entrada correspondiente;
- Las corridas de prueba, con los comentarios pertinentes;
- Este enunciado.
- Un CD conteniendo todo el material digital. Por favor no pegarle etiquetas.

9. Fechas

- Fecha de entrega: Jueves 27 de Abril.

Referencias

[1] GXemul, http://gavare.se/gxemul/.

[2] The NetBSD project, http://www.netbsd.org/.

[3] Debugging with GDB, https://sourceware.org/gdb/onlinedocs/gdb/

[4] Linux objdump command examples, http://www.thegeekstuff.com/2012/09/objdump-examples/

[5] GCC 3.3.6 Manual, https://gcc.gnu.org/onlinedocs/gcc-3.3.6/gcc/

[6] Gera's Insecure Programming (mirror), https://github.com/deadbits/InsecureProgramming