

1. LLAMADAS AL SISTEMA

El Sistema Operativo le da el control a ciertas aplicaciones para que estas puedan realizar acciones privilegiadas para ejecutarse adecuadamente y cumplan la función por la que fueron programadas. También son llamadas LLAMADAS AL SUPERVISOR.

2. NIVELES DE EXCEPCIONES

ARM consta de 4 niveles de excepciones (EL) que se enumeran del 0 [EL0] al 3 [EL3]. Donde 0 es para menores privilegios de aplicación y 3, el cual consta de permisos más privilegiados.

5. PROCEDIMIENTO DE UNA LLAMADA

Al momento de hacer una llamada al sistema en un programa tan simple como lo es un HOLA MUNDO, se hace lo siguiente:

1. Se establecen los 32bits de propósito general con la respectiva excepción.
2. Se llama a la instrucción SVC
3. Normalmente cuando el kernel implementa la función SYS, la biblioteca Glibc se encarga de hacer la llamada

LLAMADAS AL SISTEMA (LINUX) EN ARM64

3. TABLA DE VECTORES DE EXCEPCIÓN

Las excepciones están guardadas en tablas llamadas de vectores de excepción, las cuales contienen código. Cada nivel de excepción tiene su respectiva tabla y están almacenadas de forma contigua.

De esta forma, se tiene una dirección base en la siguiente dirección: `arch/arm64/kernel/entry.S +259`
El manejador se encarga de encontrar la dirección base y de acuerdo a la excepción, este se desplaza a la dirección indicada que contendrá un conjunto de 32 instrucciones de largo.

4. (CVS)

Existe una llamada instrucción CVS la cual es capaz de aumentar o disminuir los privilegios de la aplicación con la ayuda de la interpretación del kernel. Con modificar los privilegios, nos referimos a, por ejemplo, aumentar de un EL0 a un EL1.



```
int main()
{
    printf("Hello world!\n");
    return 0;
}
```

Imagen 1. Programa "Hola mundo" escrito en C

GLIBC

```
1 |0x7fb7f407a8 <__GI___libc_write>    stp    x29, x30, [sp, #-48]!
2 >|0x7fb7f407ac <__GI___libc_write+4>  adrp   x3, 0x7fb7fd1000 <__libc_pthread_f
3 |0x7fb7f407b0 <__GI___libc_write+8>   mov    x29, sp
4 |0x7fb7f407b4 <__GI___libc_write+12>  str    x19, [sp, #16]
5 |0x7fb7f407b8 <__GI___libc_write+16>  sxtw   x19, w0
6 |0x7fb7f407bc <__GI___libc_write+20>  ldr    w0, [x3, #264]
7 |0x7fb7f407c0 <__GI___libc_write+24>  cbnz   w0, 0x7fb7f407f0 <__GI___libc_writ
8 |0x7fb7f407c4 <__GI___libc_write+28>  mov    x0, x19
9 |0x7fb7f407c8 <__GI___libc_write+32>  mov    x8, #0x40
10 |0x7fb7f407cc <__GI___libc_write+36>  svc    #0x0
11
```

Imagen 2. Instrucciones en ensamblador que arroja LIBC__WRITE

PROCEDIMIENTO

El programa se compila y se exporta a alguna arquitectura ARM64.

Se ejecuta con la ayuda de GDBSERVER y se conecta remotamente. Finalmente, la última llamada a CVS arroja las instrucciones de LIBC__WRITE

INSTRUCCIONES

Se carga la dirección de la tarea que se ejecutará SINGLE_THREAD_P

Mueve el STACK-POINTER actual al FRAME_POINTER

Guarda valores en la pila para que no se pierdan al hacer el BACKUP W0 en X19. Ya que, al tener tres parámetros, estos se guardarán en x0, x2 y x3.

Carga el global en W0. W0 contendrá si el programa es multihilo o no, para después pasarlo a una condicional. En caso de que si, hace un salto pero nuestro HOLA MUNDO no lo es, así que sigue.

Finalmente, carga en x8 el valor que el Kernel proximately obtendrá para la llamada, en este caso es 64. 64 es la llamada al sistema para ESCRITURA. Al obtener esto, se llama a SVC y el kernel vuelve a tomar el control.

Referencias

1. Kannan, B. (2021, July 31). Anatomy of Linux system call in ARM64. East River Village. <https://eastrivervillage.com/Anatomy-of-Linux-system-call-in-ARM64/>

Integrantes:

- **Martinez Licea Christian Jair**
- **Santiago Alejandro Aldo**