Sri Lanka Institute of Information Technology

Systems and Network programming

IE2012

YEAR 2 SEMESTER 1

DIRTY COW EXPLOITATION

NAME: S.A.T.E SENEVIRATHNE

REG NO: IT19032870

INTRODUCTION

Linux is one of the powerful operating systems or a kernel as Windows and MacOS distributed as an Open source license. In fact, most of the softwares which are being used are developed with the use of this Operating System. Compared to other type of Operating systems which are available, Linux is very scalable and cost effective. It is also the fastest performing operating system for we hosting. It has several number of distributions such as CentOS, ubuntu, Redhat, Fedora, kali and etc. These distributions are very similar except for the few changes in the layout, configuration settings, update mechanisms, and bundled configuration. No matter what operating system it is, all of these operating systems uses a kernel. It grunts the work of the operating system. It is also called the heart of the operating system because it manages the tasks between the operating system and the hardware via the drivers installed to the kernel. It is involved highly in resource management. It optimizes the usage of the processor to make the process happen faster. It also guards against deadlocks, which are problems that completely halt the system when one application needs a resource that another application is using.

However this Operating system contains countless vulnerabilities which allows hackers to perform several exploitations easily within the operating system itself. The Race condition technique, is a Linux kernel vulnerability that misuses the mechanism used by Linux kernel to avoid deadlock to implement the Dirty COW attack on a Linux based systems. The rest of the review will provide a proper understanding about this vulnerability.

DIRTY COW (COPY ON WRITE)

The Dirty cow vulnerability of privilege escalation which allows an unprivileged local user to modify files, bypassing the standard permission mechanisms thus leading to privilege escalation on the system. Once the attacker has control over the system as a normal user, he can use an exploit which uses this vulnerability to gain full control of a Linux system and can install malware and steal data etc.

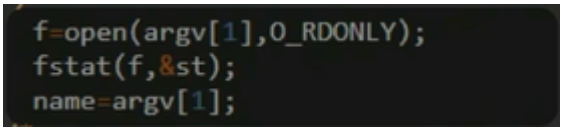
And another problem with Dirty COW is that it is almost impossible to detect by Antivirus or any security software and no evidence left for the actions taken once it is exploited. As most of the softwares are being developed using Linux it needs to be patched.

Basically, dirty cow happens when someone exploits the race condition technique. Race condition vulnerability is a flaw that produces an unexpected result when the timing of action impact other actions. In an Operating system environment, operations work in multithread which means operation is completed and then the next operation starts and ended. In order to understand this one should have a proper knowledge of how Linux manages memory. what most modern operating systems including Linux Windows Mac OS Etc do is that the memory that the process uses is a sort of logical address space. It only exists to that process. the operating

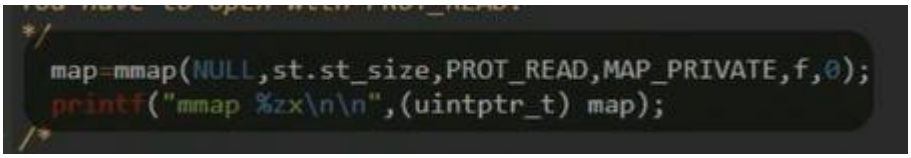
system actually thinks about it in terms of pages. These are blocks of memories which are mapped in and out as necessary. This is how you can do virtual memory and things and pass things in and out from disk when you need it. It's got its memory address space. its a flat Memory address space which you can write into in these Pages. what the operating systems doing on the other hand is its thinking about the physical memory and with the help of the CPU. It's got a translation table with a Logical view of the process and the actual physical memory. Inside the CPU there's a translation table which is set up by the operating system when this process is in memory that maps these pages into the locations. Now one of the things you can then do is say well actually say this page here has exactly the same data in it as another page in a different process. The problem comes when this process wants to modify something in that memory location. In that case what happens is the Linux kernel detected is modifying it so it's no longer the same between the two processes and will copy this one into the physical page and update the translation table. That's basically how the copy on write happens.

THE EXPLOIT CODE

- First, it opens the file, we want to write to as read-only.



- Next a Mmap call is used to create a new map to memory segments in the current process. This parameter [f] can be a file descriptor. In this case, it's the read-only file owned by root. This means it Maps the file into a new memory area. The permission Flags [PROT_READ] show that this memory area is read only. The map private [MAP_PRIIVATE] creates a private copy on write mapping. It doesn't copy the whole content of the file into memory but it maps the file into the memory. It doesn't take much of a RAM to load the file as well. even though the fire was mapped as read only because of the private mapping we can write to a copy of it. So the important takeaway here is that Mmap will map the root file directly into your memory and you can read the content of the file or write to a copy of it.



- Next, we start to threats that will run in parallel. Dirty cow is a race condition vulnerability. This means certain events have to occur in a special order to happen under normal circumstances. So you try to raise against the probability of it not happening, and you simply tried over and over.

```
  pthread_create(&pth1,NULL,madviseThread,argv[1]);
  pthread_create(&pth2,NULL,procselfmemThread,argv[2]);
/*
```

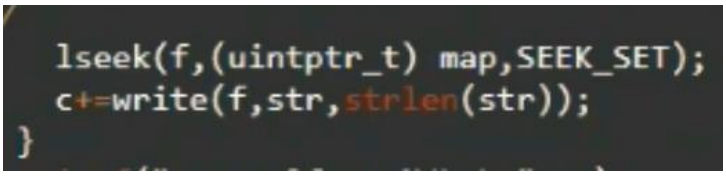
- This Thread uses a system call 'Madvice' which can be used for optimization reasons. You can provide the kernel some information on how you intend to use a memory. here the one advice we have given the kernel is that the memory area are needed [MADV_DONTNEED]

```
*/
    c+=madvise(map,100,MADV_DONTNEED);
  }
  printf("madvise %d\n\n",c);
}
```

- the other thread, Proc/self/mem opens the file proc/self/Mem. Proc is a pseudo file system. In fact most resources on Linux are managed as files. So you should always see files in quotation marks when talking about them. proc/self refers to a special file provided for the current process. So every process will have its own proc/self. And in there is a file called mem, which is a representation of the current process memory. So you could read your own processes memory from this file.

```
*/
  int f=open("/proc/self/mem",O_RDWR);
  int i,c=0;
  for(i=0;i<100000000;i++) {
/*
```

- This exploit has been written to a file in a loop. first it performs a seek [lseek] which moves the current cursor to the start of the file that that have been mapped into memory. And then advise the string we passed via the program arguments to it. this will trigger a copy of the memory so that we can write to it and see these changes. if you would do these things once, nothing would happen because that would be the expected result, but because there is a race condition issue trying this over and over again will create a weird edge case that usually doesn't occur.



CONCLUSION

So accordingly, on can perform this exploitation and gain root access on a vulnerable Linux system. This vulnerability was found by Phil Oester and this has been there since 2007. This bug is a serious vulnerability because it's so widespread, and if the conditions mentioned above are right, it gives an attacker full control over your system to install malware and steal data. And all this can be done through a very simple exploit code. The main problem is, until this vulnerability was found, software engineers has been using Linux system for different purposes. So if one was able to perform this exploitation, the whole organization will be in trouble. The patches for this has been released now for the upcoming updates. following version of kernels has been patched with the vulnerability.

3.2.0-113.155
3.13.0-100.147
3.16.36-1+deb8u2
4.4.0-45.66
4.8.0-26.28
3.10.0-327.36.3
2.6.32-642.6.2
2.6.18-416
3.0.101-84.1
3.12.60-52.57.1
3.12.62-60.64.8.2

[1] While the risk is very significant, the impact on ordinary users isn't very high due to the difficulty to get the exploit code on the systems. In terms of web services and other network connected devices, delivering the code would be difficult to do. The real risk is when user-level access exists on a device, as well as the ability to execute programs on the device. In order to avoid this happening in future, the Linux operating systems which are using old vulnerable kernels should be replaced.

Reference :

SecurityMetrics. 2020. *The Dangers Of The Dirty Cow Vulnerability: Should You Be Worried?*. [online] Available at: <https://www.securitymetrics.com/blog/dangers-dirty-cow-vulnerability-should-you-be-worried> [Accessed 12 May 2020].