Dirty Cow Vulnerability Exploit- Linux System

Shayideep Sangam

Project Report

CSC 8370 -- Data Security

Dr Robert Harrison

December 4, 2018

Abstract

In mid-2016, CVE-2016-5195 (Common Vulnerabilities and Exploits) or 'Dirty Cow' is patched. Dirty Cow is a Linux kernel race condition, which can lead to local privilege escalation. Which means a non-root user can use this exploit in a vulnerable system (Linux based system) can get root access (unauthorised) [1]. It allows a non-privileged user to take complete control over the network. This bug violates admin authorisation of security protocol. It will enable writing to protected file/memory in the system. This exploit is severe due to its massive device coverage such as all Android phones; all Linux systems are vulnerable to this exploit. The vulnerability is utilised to gain root access in any Android devices. All devices build on Linux system as base building are vulnerable to this exploit. It is present for 11 years including devices such as Ubuntu, CentOS, RedHat and Debian. Implementing this vulnerable in systems is straightforward. It is a first security issue in Linux, which is transparently fixed by official live path service. [2]

Just recently dirty cow is used in one of the security attacks against Drupal site owners. Their main aim is to gain a foothold on web servers. They used the dirty cow to increase their access to the root account, and then install a legitimate SSH client so that they can use it for future login. They do this by scanning servers in the web, which uses old Linux servers. [3]

Introduction

Linux Kernel

The kernel is the most crucial part of the Operating System that manages CPU resources, hardware and processes. [1,2] It is the lowest layer above the CPU. It is the kernel work to schedule all operations, allocate proper CPU usage to each process. Most of the system processes that interact with keyboard, mouse and other hardware devices should go through the kernel.

Copy-on-write

Copy-on-write is a technique in Linux filesystem. In this technique, the parent pages are shared with child pages when fork () calls. So, when a document is modified a separate copy of the file is not made for that process until it is written into the documents this is called Copy-on-write (COW). The newly created copy is used for the modification, not the old one. The other process continues to use the original copy of the documents for its process. [5]

Overview of the attack

This attack implemented by creating a private copy of the read-only file. It allows a malicious actor to tamper read-only and root-owned executable. Now we need to write to the private copy. As this is the first time writing to the private copy, copy-on-write comes into the picture. We need to understand the sub-process of write operation there exist a problem there.

Write operation happens in two steps:

1) Locate Physical Address

2) Write to Physical Address

In the middle of this write steps our code executes, it throws away the created private copy using madvice and executes our code. This makes kernel to write the data to the original file accidentally. [1,2]

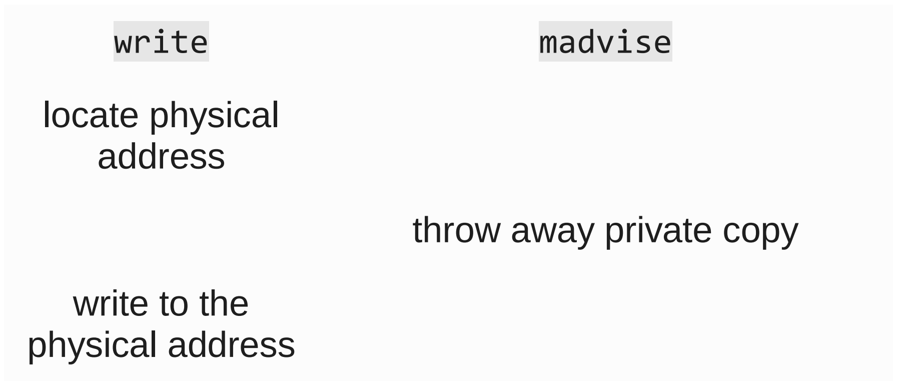
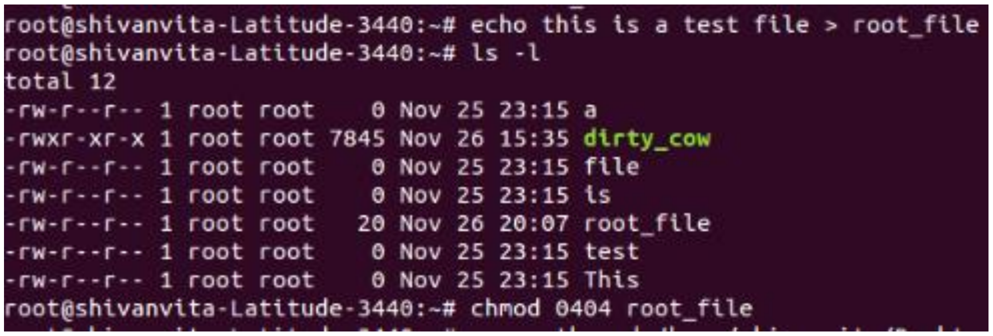


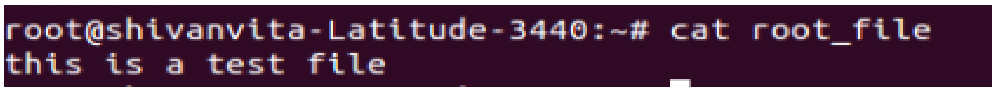
Fig1: Flow of Dirty Cow exploit implementation

Implementation

In Linux, the file system consists of root and users. The files created by root are read-only to regular users, but not write to them. No one can write the data in read-only permission. These files permission need to be changed to write-only and then written into them. Let us create a file named **root_file** in our system and write, "This is a test file" into the root_**file**. Now we have to change the data in read-only by using **chmod 0404 root_file**. Fig 2 shows the following implementation.



```
root@shivanvita-Latitude-3440:~# echo this is a test file > root_file
root@shivanvita-Latitude-3440:~# ls -l
total 12
-rw-r--r-- 1 root root    0 Nov 25 23:15 a
-rwxr-xr-x 1 root root 7845 Nov 26 15:35 dirty_cow
-rw-r--r-- 1 root root    0 Nov 25 23:15 file
-rw-r--r-- 1 root root    0 Nov 25 23:15 ls
-rw-r--r-- 1 root root   20 Nov 26 20:07 root_file
-rw-r--r-- 1 root root    0 Nov 25 23:15 test
-rw-r--r-- 1 root root    0 Nov 25 23:15 This
root@shivanvita-Latitude-3440:~# chmod 0404 root_file
```



```
root@shivanvita-Latitude-3440:~# cat root_file
this is a test file
```

Fig 2: Creating root_file in Ubuntu 14.04

Now we have to change the user permission of root_file to 0404 (read_only) and execute the dirty_cow.c program and run results to write the root_file.

madivse gives advice/suggestions to the kernel on the managing kernel sub-memory management system. Its syntax is as follows-

Int madvise (void* addr, size length, int advise);

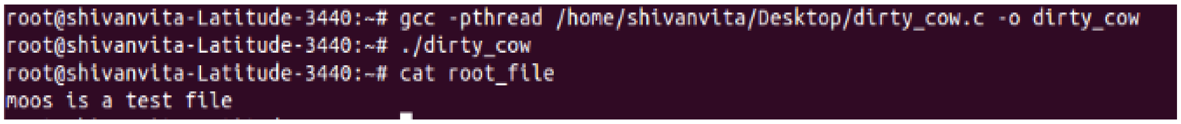
MADV_DONT NEED does not give excess shortly. Now our code runs in place of this and free resources associated with it.

(/proc/self/mem) Is a representation of our dirty_cow.c virtual memory. The Dirty Cow vulnerability requires us to use proc/self/mem because the vulnerability lives insides the Linux kernel's implementation of **process-to-process virtual memory access. [2]**

```
void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
```

By using **mmap**, refer to the file in the physical memory. We can manually provide this using mmap function given above. You have to race **madvise**(MADV_DONTNEED). [5,6]

Now we want to edit the root_file, so kernel makes an own private map for us to write to it. The Dirty cow is a race condition in Linux. In the write condition, the two processes run as two separate threads. After execution of the first thread referring to the writable file our **dirty_cow.c** run and advise the kernel to ignore the private mapping. This makes dirty_cow.c to copy given text to root_file (original). Fig 3- shows the running of dirty_cow.c code and output screenshot. [5]



```
root@shivanvita-Latitude-3440:~# gcc -pthread /home/shivanvita/Desktop/dirty_cow.c -o dirty_cow
root@shivanvita-Latitude-3440:~# ./dirty_cow
root@shivanvita-Latitude-3440:~# cat root_file
moos is a test file
```

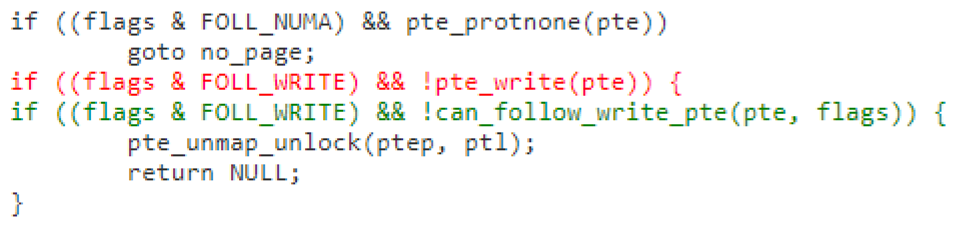
Fig 3-Shows running of dirty_cow.c and Output Screenshot

This exploit does not leave any traces in the system logs, so its tough to detect system attached using this vulnerability. This attack can be used with various other security attacks such as SQL injection, RootKit attack to intensify the attack. Due to this attack complexity, it’s difficult to differentiate attack and normal user. [6]

Fix- Dirty Cow Exploit

This attack can be used in different layers of the kernel system. Anti-Virus can be developed to detect this attack. [1,2,7] Due to this attack complexity, it's challenging to differentiate attack and regular user. This means we can program an antivirus to detect the attack. However, an attacker can change the way of using this attack by combining it with other attacks. To completely block it we need to prevent all binaries in the Linux system. Linux kernel needs to be patched followed with a full reboot. [10]

To fix this, we use a FOLL_COW to mark the execution of the copy-on-write command. We need to make sure copy-on-write is executed once, and no other program was injected in between both its operations. Below, the code shows the implementation of the FOLL_COW method.[9]



```
if ((flags & FOLL_NUMA) && pte_protnone(pte))
        goto no_page;
if ((flags & FOLL_WRITE) && !pte_write(pte)) {
if ((flags & FOLL_WRITE) && !can_follow_write_pte(pte, flags)) {
        pte_unmap_unlock(ptep, ptl);
        return NULL;
}
```

Precautions

In the Linux kernel, all file system operations are written in Object-oriented programming (OOP) style. One common abstract interface struct_file_operations can cause specific file operations. [4] Fix get_user_pages() for write access.

Impact

- An unauthorized user can access read-only files and gain write command on the system.[11]
- This flaw allows a local user to gain root access and gain complete control over the Linux systems/servers. [11]
- This flaw applies to all Android devices up to Android version 7 because Android developed over the Linux kernel. [8]
- Just recently dirty cow exploit used with another exploit to gain access of Drupal web servers. Attackers used it to get root access in web server and then installed SSH client for future login. [3]
- An attack can use this attack on old Linux based web servers and Android devices to collect user information.
- This flaw can cause easily compromise the confidentiality, integrity and availability of resources.
- Allow local users to gain privileges over Linux system. [13]
- Allow user from an off-site location to cause a denial of service. [13]

References

1) " Dirty COW and why lying is bad even if you are the Linux kernel"

(24 May 2017). Web. (18 Nov 2018).

<https://www.cs.toronto.edu/~arnold/427/18s/427_18S/indepth/dirty-cow/demo.html>.

2) "CVE-2016-5195: Dirty COW explained." (23 Oct 2016). Web. (25 Nov 2018).

Martijn Libbrecht. Retrieved from

< https://www.martijnlibbrecht.nu/2/CVE-2016-5195:_Dirty_COW_explained>.

3) Tasnim news " New Type of Attack Launched by Hackers to Take Over Web Servers Using Two Exploits.”. Web. (25 Nov 2018). Retrieved from <https://www.tasnimnews.com/en/news/2018/11/20/1880111/new-type-of-attack-launched-by-hackers-to-take-over-web-servers-using-two-exploits>.

4) “ Dirty COW and why lying is bad even if you are the Linux kernel.”

(24 May 2017). Web. (17 Nov 2009).

< https://chao-tic.github.io/blog/2017/05/24/dirty-cow>.

5) Stéphane Chazelas. “How does copy-on-write in fork() handle multiple fork?”

(6 Jan 2013.) Web. (17 Nov 2018).

<https://unix.stackexchange.com/questions/58145/how-does-copy-on-write-in-fork-handle-multiple-fork>.

6) Brain Sturk "The Dirty Truth About “Dirty COW”(CVE-2016-5195) December 7, 2016. Web. (17 Nov 2018).

< https://www.carbonblack.com/2016/12/07/dirty-truth-dirty-cow-cve-2016-5195/>.

7) Clearlinux.org “Demonstrating the Dirty Cow exploit” 18 Sep, 2017. Web. (18 Nov 2018.) < https://01.org/developerjourney/recipe/demonstrating-dirty-cow-exploit>.

8) Shaun Nichols " Dirty COW explained: Get a moooo-ve on and patch Linux root hole." (20 May 2008). Web. (25 Nov 2009.)

< https://www.theregister.co.uk/2016/10/21/linux_privilege_escalation_hole/>.

9) " mm: remove gup_flags FOLL_WRITE games from __get_user_pages()”

(13 Oct 2016). Web. (19 Nov 2018).

<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=19be0e affa3ac7d8eb6784ad9bdbc7d67ed8e619>.

10) " Dirty COW (CVE-2016-5195) is a privilege escalation vulnerability in the Linux Kernel” Web. (28 Nov 2018).

< https://dirtycow.ninja/>.

11) Nick Wilfahrt “Vulnerability Details”. Web. (27 Nov 2018)

< https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails >

12)“25 Years Of Linux - Dotdashes” Web. (28 Nov 2018) https://www.dotdashes.com/25-years-linux/

13)“Severity Ratings - Red Hat Customer Portal” Web. (November 28, 2018) <https://access.redhat.com/security/updates/classification>