Linux Kernel OS Local Root Exploit

Saleel A.P– DMC,
IT Faculty, HCT
Dubai, United Arab Emirates

Mohamed Nazeer
IT Faculty, HCT – RAK
Ras Al Khaimah, United Arab Emirates

Babak D. Beheshti, PhD,
NYIT, Old Westbury, New York

Abstract— **Dirty Copy on Write (COW) vulnerability, discovered by Phil Oester on October 2016, it is a serious vulnerability which could escalate unprivileged user to gain full control on devices (Computers, Mobile Smart Phones, Gaming devices that run Linux based operating systems). This means that any user who exploits this bug, would escalate his/her privileges; and can do anything either locally or remotely (with some modifications) to hijack the device, destroy data, create a backdoor, or to record all key strokes, use computer as an attack (object) to attack other computer in the internet (in the wild), etc.,**

COW is a local root exploit in Linux causing vulnerability issues. This paper will discuss the copy on write issue in Linux, it will also explain the nature of the problem and how it is caused, and the different mechanism to mitigate it.

Keywords- Dirty COW; Race Condition; CVE CVE-2016-5195; Linux Kernel; Local Root Exploit; OS.

I. INTRODUCTION TO NATURE OF ATTACK

COW (Dirty Copy on Write) is a security vulnerability in Linux Kernel, which affects all Linux based operating systems including Android; to gain write permission in a read-only memory mapping. This enables unprivileged local user to gain full control permission such as root account by exploiting the very old race condition of a copy on write procedure, which was discovered by Phil Oester [17]. This exploit was recorded under the reference CVE- 2016-5195. CVE that stands for Common Vulnerabilities and Exposures website managed by MITRE, is a dictionary of common names for publicly know cybersecurity vulnerabilities. [2]

Specifically, when one opens a file, its location is mapped to a certain memory space. One is not allowed to write to this location. The Linux kernel has to copy it to a new memory location before it allows writing to an underlying file.

If one has a file for which one does not have a write permission (meaning that one can read the file but not write), then this vulnerability exploits a race condition, in which an attacker may arrange to write a copy in new memory location, trying this repeatedly, until ultimately an underlying actual file write will be accomplished.

Linux is a multiprocessing and multithreaded operating system. This means that at any given point in time, if there are more than one process or thread to be run, then the processes or threads are assigned a to a queue, in which they will be allowed to run in a specific time and sequences. Linux manages all of its resources as files.

Process is a program, which is currently running in an operating system. Thread in an operating system is a particular set of instructions from a program, which is scheduled to be executed. A thread is a stream of instructions that shares the memory and other resources of a process. Therefore, a thread is a component of a process.

Race condition occurs when different processes or threads accessing the same-shared resources, such as memory, at different timing or sequence, tend to overlap between their execution timing and sequence. That is, before executing an instruction from one process, an instruction from another different process executes. By this kernel behavior, a memory is flagged as writable before it is copied (COW), and the exploit takes advantage of writable marked memory location and makes changes, circumventing the standard file permission [1]. Race condition can be caused in several ways. For example, it can be caused by multiple processes or files concurrently reading and writing on a shared memory location, where threads are racing to get the time to complete the task. Another way is to execute instructions out of sequence [15].

A vulnerability is defined as a weakness in a computer system, which an attacker exploits to compromise the security of a computer system, or to take control of the computer system. [14] Exploiting these vulnerabilities, that is the way Linux manages resources as file, handling of memory subsystem by kernel and the race condition issue, could allow an attacker to gain the root permission (this user account has complete control on a system). An attacker can then overwrite the “passwd” file in Linux, and escalate attacker privileges. [17] [12]

The Linux kernel is a monolithic Unix-like operating system kernel. Linux is highly prevalent in both traditional computer systems (such as personal computers) and servers, usually in the form of Linux distributions. Linux is also present in various embedded devices such as routers, wireless access points, PABXes, set-top boxes, FTA receivers, smart TVs, PVRs and NAS appliances. The Android operating system for tablet computers, smartphones and smartwatches is also based on the Linux kernel [9]

II. Exploit Details And How It Does It

Any user can use this vulnerability to perform a variety of attacks. It is listed under this Proof of Concept reference [19]. In the following, we will look into how an unprivileged local user can use this vulnerability to gain write permission for the files which he/she has read permission. This bug also allows an attacker to modify binaries on disk by circumventing standard permission: (Wilfahrt, 2016)

A look into the Dirtycow.c code published in the given reference (dirtycow.c code, 2016) can be useful. The following code opens a file in read only mode for which root has the write permission and we (unprivileged user) want to this file.

```
f=open(argv[1],O_RDONLY);

fstat(f,&st);

name=argv[1];
```

Next, the following interesting code, mmap, is used to create a new mapped memory space for a current process, which maps a file into memory and marks it as read only with the descriptor "f" marked red in the following code. Another important flag is Map_Private which creates private copy on write mapping (COW).

```
map=mmap(NULL,st.st_size,PROT_READ,

MAP_PRIVATE,f,0);

printf("mmap %zx\n\n",(uintptr_t) map);
```

Mmap does not copy a file into the memory. Rather it just creates a mapping to an actual file in the hard disk, to improve the operating system's performance. Therefore, if we want to write to the read only file, which is mapped to a memory location, Linux kernel would make a copy of it to do so. The important point to note in here is that mmap will map the root privileged file into a memory space, not the actual file, and you can read the file or write to a COPY of it. Changes to the copy of the root file should not be applied to the actual root privileged file.

COW starts two threads which will run concurrently to create a race condition as explained earlier. In normal conditions, certain execution of threads' instruction happens in a timely and specific order, where each processes are mutually exclusive.

```
pthread_create(&pth1,NULL,madviseThread,argv[1]);

pthread_create(&pth2,NULL,procselfmemThread,

                              argv[2]);
```

An intentional race condition is introduced by the following threads; which would ultimately allow a normal user to gain root (administrative) privileges. The first thread is the "madvise" thread. This thread advises the Linux kernel that actually one will not use this memory location any time soon. This is done by the DONTNEED flag.

```
c+=madvise(map,100,MADV_DONTNEED);
```

A second thread, "proctselfmemthread", within the same process, opens /proc/self/mem memory resource file; which allows a particular process (/proc/self) to open its mapped virtual memory location as file (/proc/self/mem). This allows a process read its memory by reading this file.

By the following code the exploit repeatedly seeks to write to this file in a loop. First line of this code performs a seek which takes the seek to the beginning of the file that is linked by memory mapping and the second line performs write based on the program argument we pass into it.

```
lseek(f,(uintptr_t) map,SEEK_SET);

c+=write(f,str,strlen(str));
```

This would trigger the Linux kernel to perform Copy on Write method to create a copy of this memory and write the changes to the copy not to the actual memory mapped file.

By repeating these two threads in a loop, spamming madvise thread (read) and /proc/self/mem (write), a race condition is created that tricks the Linux kernel handling the timing and sequence between these two process, causing a write to the actual file in the hard disk (which is read only file for an unprivileged user). This vulnerability in Linux kernel would allow normal user to gain root privileges by easily exploiting it to hijack any Linux based devices like Android, Personal computers, servers, gaming devices etc., (Explaining COW local root exploit - CVE-2016-519, 2016) (Nichols, Widespread flaw can be easily, n.d.). According to Phil Oester, discoverer of COW, in Dan Goodin's explanation "Any user can become root in less than 5 seconds in my testing, very reliably. Scary stuff." (GOODIN, 2016)

Alternatively, as explained by Shaun Nichols (Nichols, Widespread flaw can be easily exploited to hijack PCs, servers, gizmos, phones, 2016), COW happens because, due to a race with madvise(), the kernel does not fully break the executable from the process's private memory. The process writes to the read-only object, triggers a page access fault, and the fault is handled by allocating a new page containing a copy of the underlying file and privately mapping it into the process. This is fine by itself. However, madvise() tells the kernel to discard the pages holding the mapped area. Calling this while writing to /proc/self/mem will eventually lead to an inconsistent state where the pages holding the mapped executable are removed; and a write is attempted to the memory area. A write that should go to the private pages will instead alter the mapped object. These changes are committed by the kernel to storage.

This can be exploited to alter a root-owned setuid binary so that it, for example, spawns a root-owned shell. It is possible to combine this exploit with ptrace to make it more reliable, although it is not essential.

III. How It Was Spread And Identified

An article published by the beta news claims that a vulnerability discovered in the Linux kernel nine years ago. Discovered by security expert Phil Oester, Linux users were advised to seek out and install a patch as soon as possible. Dubbed COW, this bug is identified as a privilege escalation vulnerability, which can be found in just about every Linux distribution. (Wilson, 2016)

COW is termed as one of the most serious and alarming bugs of its type ever found in Linux. There are many evidences that this vulnerability is been exploited since then The CVE has classified it as CVE-2016-5195 code. A website is already set up to alert people to the problem and advises that the "security community should deploy honeypots that entrap attackers and to alert about exploitation attempts".

The Linux patches for this bug are already released. It is important that Linux users to ensure that the patches are installed on their system., The potential impact of a successful exploit is huge, as majority of the web servers around the world are subscribers of the Linux operating system. This exploit is a major concern now for Linux customers, as it is almost impossible for antivirus and security software to detect it.

Any web server/application vulnerability which allows the attacker to upload a file to the impacted system and execute could also make it work. (Wilson, 2016)

Oester claims that this particular exploit uploaded to his system was compiled with GCC 4.8.5 released 20150623. This should not imply that the vulnerability was not available earlier than that date, given its longevity., Anyone running Linux on a web facing server is vulnerable to this attack and is being targeted. He has been capturing all inbound traffic to his webservers for forensic analysis. This practice has proved invaluable on numerous occasions, and he recommend it to all admins.

IV. Mechanisms To Counteract

Although the attack can happen in different layers, antivirus signatures that detect COW could be developed. Due to the attack complexity, differentiating between legitimate use and attack cannot be done easily; but the attack may be detected by comparing the size of the binary against the size of the original binary. This implies that antivirus can be programmed to detect the attack but not to block it unless binaries are blocked altogether. (Wilson, 2016)

The Red Hat distribution of Linux is aware of this issue and a fix has been released. All Red Hat customers with affected versions of the kernel are strongly recommended to update the kernel as soon as patches are available. Details about impacted packages as well as recommended mitigation are available at their website. A system reboot is required in order for the kernel update to be applied. (Kernel Local Privilege escalation - Resolved, 2016)

V. Mitigation

Red Hat Support is providing mitigations such as SystemTap scripts as a stop-gap until official patches are released. SystemTap scripts are only supported for 30 days after the GA release of the final patches.

VI. How To Build And Use The Systemtap Workaround

The systemtap countermeasure includes creating a kernel module (like a driver) using a systemtap script that intercepts the vulnerable system call. It is used as a stopgap solution until a fixed kernel is booted into the affected machine. This solution does not require a reboot and applies to RHEL 5, 6 and 7.

VII. Requirements

In order to build the systemtap module, the following packages are required:

- systemtap-client
- systemtap-devel
- gcc (and dependencies)
- kernel-devel-`uname -r`
- kernel-debuginfo-`uname -r`
- kernel-debuginfo-common-`uname -r`

The 'kernel' packages requires the same version as the running kernel. Downloading the latest version will prevent systemtap for working. Please download the exact running version.

VIII. How Do I Build The Module?

1. After you installed the packages, create a file named dirtycow.stp with this content:

```
probe kernel.function("mem_write").call ? {
        $count = 0
}
probe syscall.ptrace {  // includes compact ptrace as well
        $request = 0xfff
}
probe begin {
        printk(0, "CVE-2016-5195 mitigation loaded")
}
probe end {
        printk(0, "CVE-2016-5195 mitigation unloaded")
}
```

```
2. Save the file. Compile it using this command:
# stap -g -p 4 -m dirtycow_`uname -r|tr -cd [:digit:]` dirtycow.stp
dirtycow_26183985.ko
```

In the above example, the. ko file has a number identifying its kernel version. In that case, it is 2.6.18-398.el5. This module can be used in other systems with this exact kernel version without having to install the debuginfo and development packages, needing only systemtap-runtime. Just copy the module file to the server with the same kernel version, install the systemtap-runtime package and proceed from the next step.

3. In order to load the module, run the command staprun -L

<.ko file>. For example:

```
# staprun -L dirtycow_26183985.ko
```

4. Check if is it loaded:

```
# dmesg | grep CVE-2016-5195
CVE-2016-5195 mitigation loaded
```

5. To unload the module, reboot the system or run the "staprun -A dirtycow_26183985" command and interrupt it with Ctlr+C, as shown below:

```
# staprun -A dirtycow_26183985
^C
Message from syslogd@...
kernel:CVE-2016-5195 mitigation unloaded
```

IX. Important Notes

- The module does not survive a boot: It is not reloaded after a system boot.
- After a reboot, the module must be manually loaded again.
- If the kernel is updated or changed, this module will not be loaded into the new kernel.
- If you booted into a different kernel without the fix, a new compatible module should be loaded.
- A corrected kernel does not need the module load.

Other flavors of Linux (Red Hat. Debian. Ubuntu. SUSE) also have released the fixes for this vulnerability and the related fixes/updates could be found in their corresponding websites. Applying the fix is straightforward: Most systems will fix this by updating your system and rebooting your server.

For Example: "On Ubuntu and Debian, upgrade your packages using apt-get

```
$ sudo apt-get update && sudo apt-get dist-upgrade
```

You can update all of your packages on CentOS 5, 6, and 7 with sudo yum update, but if you only want to update the kernel to address this bug, run:

```
$ sudo yum update kernel
```

On older Droplets with external kernel management, you will also need to select the DigitalOcean GrubLoader kernel (Virdo, 2016)

X. Conclusion

Although Copy on Write functionality of Linux kernel which exist in respect to its efficient memory management based on functional aspect. However, Race condition vulnerability is not predictable and programmatically difficult to verify despite of different patches to protect against this condition. Only possibility is to protect our system or more appropriate statement is to maintain our system is by Defense in Depth; Host Hardening by keeping our operating system up to date, proper host based firewall, etc., these kind of vulnerability will always exist by a persistent intruder.

References

[1] crovers. (2016, OCT 21). Simple explanation of how Dirty COW works. Retrieved from security.stackexchange.com: http://security.stackexchange.com/questions/140469/simple-explanation-of-how-dirty-cow-works

[2] cve. (2016, SEP 13). Common Vulnerabilities and Exposures. Retrieved from cve.mitre.org: https://cve.mitre.org/about

[3] Dirty COW (CVE-2016-5195) is a privilege escalation vulnerability in the Linux Kernel. (2016). Retrieved from Dirtycow.ninja: https://dirtycow.ninja/

[4] dirtycow.c code. (2016). Retrieved from github.com: https://github.com/dirtycow/dirtycow.github.io/blob/master/dirtyc0w.c#L2

[5] Explaining Dirty COW local root exploit - CVE-2016-519. (2016, OCT 21). Retrieved from Video: youtube.com: https://www.youtube.com/watch?v=kEsshExn7aE

[6] GOODIN, D. (2016, OCT 10). Most serious" Linux privilege-escalation bug ever is under active exploit . Retrieved from arstechnia.com: http://arstechnica.com/security/2016/10/most-serious-linux-privilege-escalation-bug-ever-is-under-active-exploit

[7] ISO/IEC, 1. (2011, SEP 01). Information technology -- Programming languages -- C++. Retrieved from www.iso.org: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50372

[8] Kernel Local Privilege escalation - Resolved. (2016, 12 02). Retrieved from access.redhat.com/security: https://access.redhat.com/security/vulnerabilities/DirtyCow?page=1

[9] Linux Kernel Readme. (2010, NOV). Retrieved from Kernel.org git repositories: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/README?id=f3b8436ad9a8ad36b3c9fa1fe030c7f38e5d3d0b

[10] Nichols, S. (2016, OCT 21). Widespread flaw can be easily exploited to hijack PCs, servers, gizmos, phones. (www.theregister.co.uk) Retrieved Dec 2016, from http://www.theregister.co.uk/2016/10/21/linux_privilege_escalation_hole/

[11] Nichols, S. (n.d.). Widespread flaw can be easily. Retrieved from www.theregister.co.uk: http://www.theregister.co.uk/2016/10/21/linux_privilege_escalation_hole/

[12] Oester, G. B.-P. (2016, OCT 20). Linux users urged to protect against 'Dirty COW' security flaw. Retrieved from www.v3.co.uk: http://www.v3.co.uk/v3-uk/news/2474845/linux-users-urged-to-protect-against-dirty-cow-security-flaw

[13] Paganini, P. (2016, OCT 21). The new dirty COW Linux kernel exploit already used in attacks in the wild. Retrieved from The new dirty COW Linux kernel exploit already used in attacks in the wild: http://securityaffairs.co/wordpress/52521/hacking/dirty-cow-exploit.html

[14] The Three tenets of Cyber Security. (2009, DEC 15). Retrieved from U.S. Air Force Software Protection Initiative: https://www.spi.dod.mil/tenets.htm

[15] Unger, S. H. (1995, JUN). Hazards, critical races, and metastability. Retrieved from IEEE: http://ieeexplore.ieee.org/document/391185/?reload=true&tp=&arnumber=391185

[16] Virdo, H. (2016, 10 2016). How To Protect Your Server Against the Dirty COW Linux Vulnerability . Retrieved from www.digitalocean.com: https://www.digitalocean.com/community/tutorials/how-to-protect-your-server-against-the-dirty-cow-linux-vulnerability

[17] Wilfahrt, N. (2016, OCT 30). Dirty Cow VulnerabilityDetails. Retrieved from github.com: https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails

[18] Wilson, M. (2016, Dec 12). Serious Dirty COW bug leaves millions of Linux users vulnerable to attack. Retrieved from betanews.com: http://betanews.com/2016/10/22/dirty-cow-linux-vulnerability/

[19] Yoshi, T. (2016). Dirty CoW - Proof Of Concepts. Retrieved from dirtycow/dirtycow.github.io: https://github.com/dirtycow/dirtycow.github.io/wiki/PoCs.