Dirty COW

by Jake Wilson and Nimesha (Nim) Jayawardena

A visually explained demo can be found here.

Dirty COW was a vulnerability in the Linux kernel. It allowed processes to **write to read-only files**. This exploit made use of a race condition that lived inside the kernel functions which handle the copy-on-write (COW) feature of memory mappings. An example use case includes over-writing a user's UID in `/etc/passwd` to gain root privileges. Dirty COW is listed in the Common Vulnerabilities and Exposures as CVE-2016-5195.

The vulnerability had existed in the Linux kernel since 2007. It was discovered and partially patched in 2016 (and fully patched in 2017).

What is the Linux kernel?

The kernel of an operating system is the piece of software that handles all other processes and hardware. If you run some process that writes to a file, your process — under the hood — talks to the kernel, "hey, write this to that file." The kernel also manages the scheduling of processes, making sure every process gets to use the CPU. Additionally, when your process interacts with keyboards, microphones, and other hardware, your process must go through the kernel.

During the 1980s, there was virtually no "free" OS (operating system). If you wanted to play around with an OS (e.g., for educational purposes), you had to resort to strictly

licensed OSes like Minix. In 1983, a man named Richard Stallman started the GNU Project with the goal of creating completely free software. Although, the GNU Project resulted in the creation of many software needed for a complete OS (such as a compiler and a shell), the project's production of a kernel was delayed. Frustrated by the lack of **free** OS kernels, Linus Torvalds created the Linux kernel in 1991.

What is COW?

When a process requests a copy of some data (e.g., a file), the kernel does not create the actual copy until it's being written into. This technique is called copy-on-write (COW).

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 `madvise`. This throwing away of the private copy results in the kernel accidentally writing to the original read-only file.

`write`	`madvise`
locate physical address	
	throw away private copy

`write`	`madvise`
write to physical address	

For a more detailed visual explanation, check out our demo page. It'll be fun!

Resources

- Jake's Slides (.pptx)
- In-depth Kernel Level Walkthrough (Article)
- Dirty COW Explained (YouTube)