APPLE AND MICROSOFT PUT A BACK-DOOR ON EVERY COMPUTER!!!!

Now YOUR kernels can be hijacked or crashed. MS and APPLE say it was an 'accident' but few insiders believe them

Grab those patches while Chipzilla updates its manuals

By Simon Sharwood and Chris Williams 110 Reg comments SHARE ▼

Linux, Windows, macOS, FreeBSD, and some implementations of Xen have a 'design flaw' (AKA: Backdoor) that could allow attackers to, at best, crash Intel and AMD-powered computers.

At worst, miscreants can, potentially, "gain access to sensitive memory information or control low-level operating system functions," which is a fancy way of saying peek at kernel memory, or hijack the critical code running the machine.

The vulnerabilities can be exploited by malware running on a computer, or a malicious logged-in user. Patches are now available to correct the near-industry-wide programming blunders.

As detailed by CERT on Tuesday, the security cockup, labeled CVE-2018-8897, appears to have been caused by developers at Microsoft, Apple, and other organizations 'misunderstanding' the way Intel and AMD processors handle one particular special exception.

Indeed, CERT noted: "The error appears to be due to developer interpretation of existing documentation." In other words, programmers misunderstood Intel and AMD's manuals, which may not have been very clear.

You're fired (the interrupt, that is)

Here's a deep dive put as gently as possible. At the heart of the issue is the `POP SS` instruction, which takes from the running program's stack a value used to select the stack's segment, and puts that number into the CPU's stack selector register. This is all to do with memory segmentation that modern operating systems mostly ignore, and you can, too. The `POP SS` instruction is specially handled by the CPU so that the stack cannot be left in an inconsistent state if an interrupt fires while it is executing.

An application can set a debug breakpoint for the memory location where that stack selector will be pulled from the stack by `POP SS`. That is, when the app uses `POP SS`, it will generate a special exception when the processor touches a particular part of RAM to fetch the stack selector.

Now, here's the clever trick. To exploit this situation, the instruction immediately after the `POP SS` instruction has to be an `INT` instruction, which triggers an interrupt. These software-generated interrupts are sometimes used by user programs to activate the kernel so it can do work for the running process, such as open a file.

On Intel and AMD machines, the software-generated interrupt instruction immediately after `POP SS` causes the processor to enter

the kernel's interrupt handler. Then the debug exception fires, because `POP SS`caused the exception to be deferred.

Operating system designers didn't expect this. They read Intel's x86-64 manuals, and concluded the handler starts in an uninterruptable state. But now there's an unexpected debug exception to deal with while very early inside the interrupt handler.

This confuses the heck out of the kernel, causing it to, in certain circumstances, rely on data controlled by un-privileged user software, as explained by the flaw's discoverers Nick Peterson of Everdox Tech, and Nemanja Mulasmajic of triplefault.io, in their technical explanation (PDF):

> **When the instruction, `POP SS`, is executed with debug registers set for break on access to that stack location and the following instruction is an `INT N`, a pending `#DB` will be fired *after* entering the interrupt gate, as it would on most successful branch instructions. Other than a non-maskable interrupt or perhaps a machine check exception, operating system developers are assuming an uninterruptible state granted from interrupt gate semantics. This can cause OS supervisor software built with these implications in mind to erroneously use state information chosen by unprivileged software.**
>
> **This is a serious security vulnerability and oversight made by operating system vendors due to unclear and perhaps even incomplete documentation on the caveats of the POP**

> **SS instruction and its interaction with interrupt gate semantics.**

The upshot is that, on Intel boxes, the user application can use `POP SS`and `INT` to exploit the above misunderstanding, and control the special pointer `GSBASE` in the interrupt handler. On AMD, the app can control `GSBASE` and the stack pointer. This can either be used to crash the kernel, by making it touch un-mapped memory, extract parts of protected kernel memory, or tweak its internal structures to knock over the system or joyride its operations.

Any exploitation attempt is more likely to crash the kernel than cause any serious harm, we reckon. However, like Meltdown, as bugs go, it's a little embarrassing for the industry, and it ought to be patched to be on the safe side.

Manipulations

The FreeBSD advisory on the problem explains it further. “On x86 architecture systems, the stack is represented by the combination of a stack segment and a stack pointer, which must remain in sync for proper operation,” the OS’s developers wrote. “Instructions related to manipulating the stack segment have special handling to facilitate consistency with changes to the stack pointer.

“The MOV SS and POP SS instructions inhibit debug exceptions until the instruction boundary following the next instruction. If that instruction is a system call or similar instruction that transfers control to the operating system, the debug exception will be handled in the kernel context instead of the user context.”

The result? “An authenticated local attacker may be able to read sensitive data in kernel memory, control low-level operating system functions, or may panic the system.”

Exploiting such on Windows, according to Microsoft’s kernel advisory, would mean “an attacker would first have to log on to the system. An attacker could then run a specially crafted application to take control of an affected system.”

Which – gulp! - isn’t a very far-fetched scenario, unless you run a tight ship of no untrusted code.

Red Hat has patches ready to roll, as does Ubuntu, and Apple for macOS.

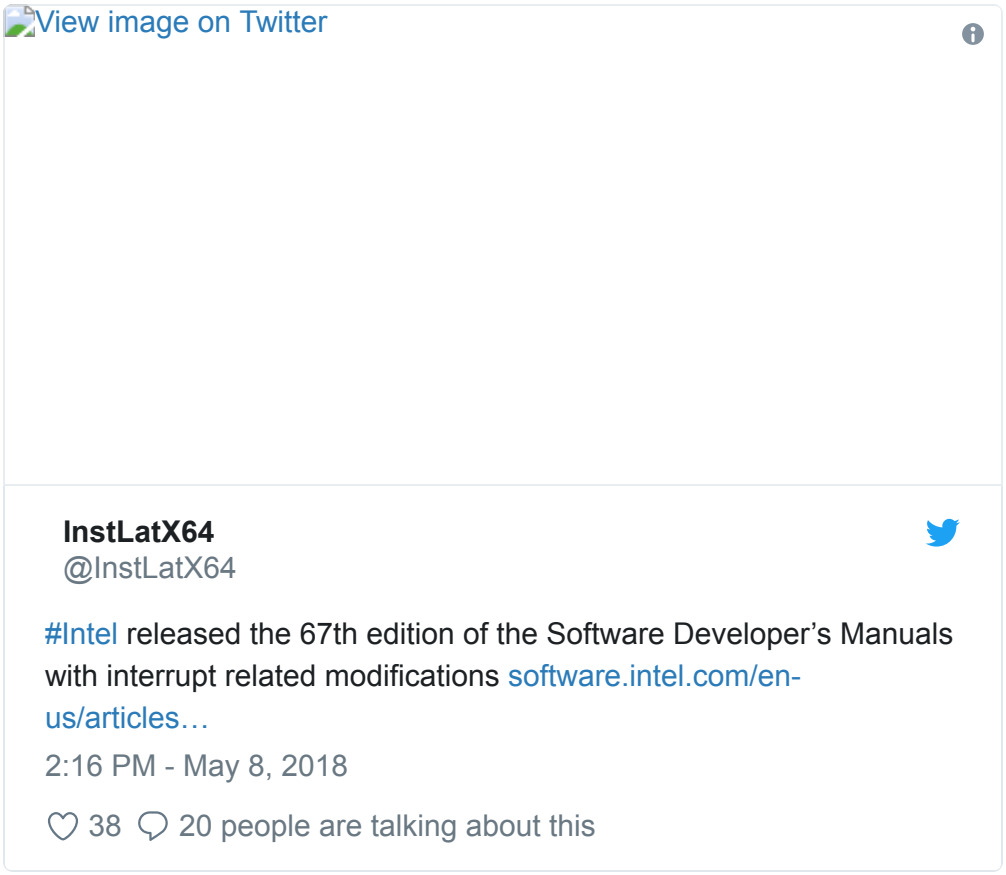
The Linux kernel has also been fixed, way back on March 23, 2018. A patch is already present in versions 4.15.14, 4.14.31, 4.9.91, 4.4.125, plus older 4.1, 3.16, and 3.2 branches.

Microsoft’s got it sorted, for Windows 7 through 10 and Windows Server 2008 through version 1803. Xen has patches for versions 4.6 through 4.10. VMware’s hypervisors aren’t at risk, but vCenter Server has a workaround and vSphere Integrated containers await a fix, but both are rated merely “potentially affected.”

See the above CERT link for all affected vendors and their responses, and apply updates as necessary.

All sources are at pains to point out that while this issue derives from an x86-64 instruction, kernel programmers, and not Chipzilla, are to blame. It seems lots of coders have simply misunderstood how to

handle debug exceptions, and made similar mistakes over a long period of time.



The Register expects plenty of OS developers are about to be sent to compulsory reeducation sessions on the x86-64 architecture, now that Intel has updated its manuals to clarify the handling of stack selector instructions, and that readers get to do the emergency patch thing. Which you should be pretty good at by now. ®