Security

Meet TLBleed: A crypto-key-leaking CPU attack that Intel reckons we shouldn't worry about

How to extract 256-bit keys with 99.8% success

By Chris Williams, Editor in Chief 22 Jun 2018 at 22:44 60 SHARE ▼

Updated Intel has, for now, no plans to specifically address a side-channel vulnerability in its processors that can be potentially exploited by malware to extract encryption keys and other sensitive info from applications.

A team of researchers at the Systems and Network Security Group at Vrije Universiteit Amsterdam, in the Netherlands, say they were able to leverage the security weakness to extract crypto keys from another running program in 99.8 of tests on an Intel Skylake Core i7-6700K desktop CPU; 98.2 percent of tests on an Intel Broadwell Xeon E5-2620 v4 server CPU; and 99.8 per cent of tests on a Coffeelake part.

Their code was able to lift a secret 256-bit key, used to cryptographically sign data, from another program while it performed a signing operation with libgcrypt's Curve 25519 EdDSA implementation. It took roughly 17 seconds to determine each of the keys using machine-learning software and some brute force, according to a paper detailing the attack, seen by The Register this week.

"The end-to-end attack time is composed of: 2ms of capture time; 17 seconds of signals analysis with the trained classifier; and a variable amount of brute-force guessing with a median work factor of 2^{13}, taking a fraction of a second," the team – Ben Gras, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida – stated in their paper.

The extraction technique is not reliant on speculative execution, and thus is unrelated to Spectre and Meltdown. Instead, it builds upon the exploitation of Intel's Hyper-Threading technology and the processor caches to leak data, which is a known security problem with its own mitigations.

TLBleed

The technique has thus been dubbed TLBleed as it targets a CPU's TLB: the translation lookaside buffer, which is a type of cache. The difference between TLBleed and previous cache-based attacks, according to the VU Amsterdam researchers, is that protections to thwart side-channel snooping on memory caches are not guaranteed to block TLB spying.

Before we get into the more technical stuff, we should stress that this is not the end of the world because, first, you need malware running on, or a malicious user logged into, your system to exploit it. Second, no one right now is leveraging the weaknesses in the wild. There are easier ways for hackers to extract data from a computer or other device, via security bugs in browsers, PDF readers, email clients, and so on.

And, third, exploiting this TLB side channel is non trivial.

However, if you are worried about cache-based attacks – such as, if you're running a virtual machine on a public cloud platform, and fear neighboring guests are trying to snoop on you – then you should be paying attention.

"Don't panic: while a cool attack, TLBleed is not the new Spectre," one of the researchers, Ben Gras, said on Friday.

Hyper what?

TLBleed relies on the use of Hyper-Threading, a technology present in most modern Intel chips. A processor can have a number of cores, two, four, eight, and so on, each separately fetching and executing code from memory. With Hyper-Threading enabled, each core can execute multiple threads, typically two, simultaneously. This means two threads run at the same time on the same core and share infrastructure within that core, such as its memory caches and TLB.

If a core is told by an operating system to run a thread in one program and a thread in another program simultaneously, it is then possible for one of the threads to spy on the other thread by watching how it accesses that CPU's private resources. From these observations, it is possible to determine the contents of RAM secret to that other program.

That means some malicious software can snoop on the cryptographic keys used by, say, an email client to sign or encrypt messages.

There are mechanisms available to prevent data from leaking this way via the memory caches, for example: Intel's Cache Allocation Technology, summarized here.

The memory caches keep copies of morsels of the system's RAM within the processor, which are faster to read from and write to than going out over the wire to external storage chips. The TLB, meanwhile, is a different kind of cache.

It is a table of numbers that convert memory addresses within each individual application's virtual memory space to the physical locations of bytes in RAM. As far as programs are concerned, they have nearly all of a system's memory to themselves. In actual fact, the operating system and processor are working overtime behind the scenes to create that illusion. Special data structures link virtual memory to physical memory, so that when two apps access what they've both stored at virtual memory address 4,096, one actually fetches from, say, physical address 81,920, and the other from 86,016.

These data structures – the processor and kernel's page tables – are large, and too much to fit entirely into a single cache. So the TLB within the core holds a small collection of frequently used memory location translations. If a virtual-to-physical lookup is performed, and it's not in the TLB, it will be brought in so subsequent translations are almost instant. This may involve pushing out a less frequently used lookup from the buffer.

By observing the processor recycling TLB slots with new translations, it is possible to work out how the other thread running on the same core is operating.

ASLR-security-busting JavaScript hack demo'd by university boffins

READ MORE →

And so just in the same way you can strategically manipulate a memory cache's contents to observe when another program refills it as it accesses its own private data, you can monitor the TLB – shared between two threads running in the same core – so that the other program ultimately accidentally spills the contents of its memory.

The team used AI – specifically, a support vector machine classifier – to identify when a program is executing a sensitive operation, such as a cryptographic function, through the TLB latencies, and read out that app's private data as a stream of bits, allowing them to reconstruct things like crypto keys. There are hurdles to overcome, such as address-space layout randomization – however, the team is confident these can be defeated in real-world attacks.

"TLBleed shows that, by monitoring hyper-thread activity through the TLB instead of caches, even with full cache isolation or protection policies in effect, information can still leak between processes," Gras told The Register, "allowing reliable cryptographic key recovery after just one capture of the TLB signal with a 98 per cent success rate."

One crucial component to TLBleed is not determining where in memory a program is reading and writing from and to, but when. The processor and kernel typically split applications into blocks of memory no smaller than 4KB, which is rather coarsely grained. So, the team instead looked at the timings of accesses, and, knowing the design of the code within the targeted cryptographic operation, could deduce the information being handled.

"We use machine learning to make sense of the data, and had to do an analysis on the access patterns in time, as opposed to accessing different areas," said Gras. "To distinguish the cryptographic key bits, we use a machine-learning algorithm, which is novel, I believe."

Response

Intel reckons existing cache-snooping countermeasures are sufficient to prevent data from leaking from one program to another via TLBleed, we understand. In effect, its engineers think code can be written and built so that it is TLBleed-resistant.

The semiconductor giant won't even request a CVE number for the discovered flaw – a unique industry-recognized ID number for identifying vulnerabilities – and declined to pay the team a HackerOne-hosted bug bounty it has on side-channel flaws in its chips.

"We regard this as a goalpost-moving move on Intel's part," Gras told us.

"The HackerOne bug bounty program run by Intel has side channels in scope. However, Intel has dismissed our report as it does not demonstrate a side-channel attack against its ‘constant time’ – its side-channel hardened – cryptographic primitives."

He added that he believes today's mitigations against cache snooping may not be enough to stop TLBleed, and that applications in the wild right now are potentially vulnerable:

> **If software is written to be perfectly side-channel resistant – control flow and data flow are constant and do not depend on the key being processed – [TLBleed] will not work, but these implementations are rare. Also, the fact that the TLB uses data accesses and not the code path is unusual enough that side-channel-resistant implementations have been found to be vulnerable to TLBleed, eg: RSA in libgcrypt.**
>
> **In short, this is a new side channel attack that current software defenses and coding practices are unlikely to be dealing with.**

A spokesperson for Intel told The Register: "Protecting our customers and their data continues to be a critical priority for us. We are looking into this feedback and thank the community for their ongoing efforts.”

Gras also fears AMD's hardware threading technology in its latest Zen processors – Ryzen, Threadripper, and Epyc – are at risk from TLBleed, as the CPU cores can also each run multiple threads simultaneously just like Intel parts. A spokesperson for AMD had no comment.

One headache with mitigating TLBleed in software – that is, recompiling applications so that they cannot be snooped on via memory translations – is that the design of TLBs varies from microarchitecture to microarchitecture. You might, say, be able to prevent an app from leaking data on a Skylake CPU, but run it on a Broadwell part, and you'll be back to square one: the software will potentially be vulnerable again.

Another mitigation is at the operating system level: the kernel scheduler could be told to prevent a core from simultaneously running two threads in two different programs, although this would penalize computers running lots of single-threaded processes. Alternatively, sensitive programs could run on ring-fenced cores away from non-sensitive processes. However, that would require a fair amount of plumbing within the kernel, and handholding by users.

In cloud environments, a hypervisor could ensure that a core never simultaneously runs threads from two different virtual machines.

Another mitigation is to simply turn off simultaneous multithreading completely, which incurs a performance penalty depending on the workloads you're running. OpenBSD has already gone down that route. Other operating systems may follow: we understand OS vendors and development teams, including those steering FreeBSD, NetBSD, and HardenedBSD, are evaluating the impact of TLBleed.

Finally, Intel could try splitting its TLB, in the same way it allowed programmers to carve up the cache, but it appears unwilling to do this.

The team's paper is due to be made public next week. Gras will give a talk at Black Hat USA in August on the vulnerability. ®

Updated to add

A spokesperson for Intel has been in touch, after publication, with a longer response to the TLBleed investigation:

> **Intel has received notice of research from Vrije Universiteit Amsterdam, which outlines a potential side-channel analysis vulnerability referred to as TLBleed. This issue is not reliant on speculative execution, and is therefore unrelated to Spectre or Meltdown. Research on side-channel analysis methods often focuses on manipulating and measuring the characteristics (e.g. timing) of shared hardware resources. These measurements can potentially allow researchers to extract information about the software and related data.**
>
> **TLBleed uses the Translation Lookaside Buffer (TLB), a cache common to many high performance microprocessors that stores recent address translations from virtual memory to physical memory. Software or software libraries such as Intel Integrated Performance Primitives Cryptography version U3.1 – written to ensure constant execution time and data independent cache traces should be immune to TLBleed.**

So far, libgcrypt has been shown to be vulnerable to TLBleed, and hopefully it can be and will be updated accordingly. Now it's up to similar cryptographic libraries – such as BoringSSL and OpenSSL which are side-channel-snooping resistant – as well as other software, to demonstrate and ensure they are immune to TLB-based leaks, too.

Final update

A spokesperson for AMD has been in touch to say none of its chips are susceptible to TLBleed:

> **Based on our analysis to date we have not identified any AMD products that are vulnerable to the TLBleed side channel attack identified by the researcher. Security remains a top priority and we will continue to work to identify any potential risks for our customers and, if needed, potential mitigations.**

Whitepapers

More from The Register

Ex-Intel exec Diane Bryant exits Google cloud
Could Chipzilla replace Brian with a Bryant?

Intel finds a cure for its software security pain: Window Snyder
Microsoft, Mozilla veteran will also handle external researcher work

Intel confirms it'll release GPUs in 2020

They sell like hot cakes so why wouldn't Chipzilla want in?

Intel's still-in-beta drone flight planning software gets update

Chipzilla is doing a little aviating of its own, we see

Intel gives Broadwells and Haswells their Meltdown medicine

Chipzilla and Oracle are working their way back through time to deliver fixes

Micron, Intel consciously uncouple 3D NAND development

Will continue to work on 3D XPoint together

Sponsored links