# Retbleed: Arbitrary Speculative Code Execution with Return Instructions

Retbleed (CVE-2022-29900 and CVE-2022-29901) is the new addition to the family of speculative execution attacks that exploit branch target injection to leak information, which we call Spectre-BTI. Unlike its siblings, who trigger harmful branch target speculation by exploiting indirect jumps or calls, Retbleed exploits return instructions. This means a great deal, since it undermines some of our current Spectre-BTI defenses.

One such defense that many of our operating systems use today is called retpoline. Retpolines work by replacing indirect jumps and calls with returns. Back in 2018, retpolines were invented to prevent unauthorized attackers from stealing information from the memory of the system using Spectre-BTI. While some were concerned about whether returns could be susceptible to Spectre-BTI attacks too, returns were deemed

impractical to exploit, and concerns were brushed off the shoulder. The primary reason being that return target predictions are, under normal microarchitectural conditions, not predicted as indirect branches.

As it turns out however, Retbleed is indeed practical to exploit, thanks to the following two insights:

- We found that we can trigger the microarchitectural conditions, on both AMD and Intel CPUs, that forces returns to be predicted like indirect branches. We also built the necessary tools to discover locations in the Linux kernel where these conditions are met.
- We found that we can inject branch targets that reside inside the kernel address-space, even as an unprivileged user. Even though we cannot access branch targets inside the kernel address-space — branching to such a target results in a page fault — the Branch Prediction Unit will update itself upon observing a branch and assume that it was legally executed, even if it's to a kernel address.

## Microarchitectural conditions to exploit Retbleed

**Intel.** On Intel, returns start behaving like indirect jumps when the Return Stack Buffer, which holds return target predictions, is underflowed. This happens upon executing deep call stacks. In our evaluation, we found over a thousand of such conditions that can be triggered by a system call. The indirect branch target predictor for Intel CPUs has been studied in previous work.

**AMD.** On AMD, returns will behave like an indirect branch regardless of the state of their Return Address Stack. In fact, by poisoning the return instruction using an indirect jump, the AMD branch predictor will assume that it will encounter an indirect jump instead of a return and consequentially predict an indirect branch target. This means that any return that we can reach through a system call can be exploited — and there are tons of them.

## Phantom JMPs

We also found that AMD CPUs exhibit *phantom jumps* (CVE-2022-23825): branch predictions that occur even in the absence of any corresponding branch instruction. Using the same technique we used to exploit Retbleed, we could omit the return instruction completely and observe branch target prediction on any given instruction. This has significant implications for the exposed attack surface and mitigations. Read our addendum to the Retbleed paper if you want to know more.

## Affected machines

We have verified that Retbleed works on AMD Zen 1, Zen 1+, Zen 2 and Intel Core generation 6, 7 and 8. Intel provides a more comprehensive list of its affected CPUs here and AMD provides it here.

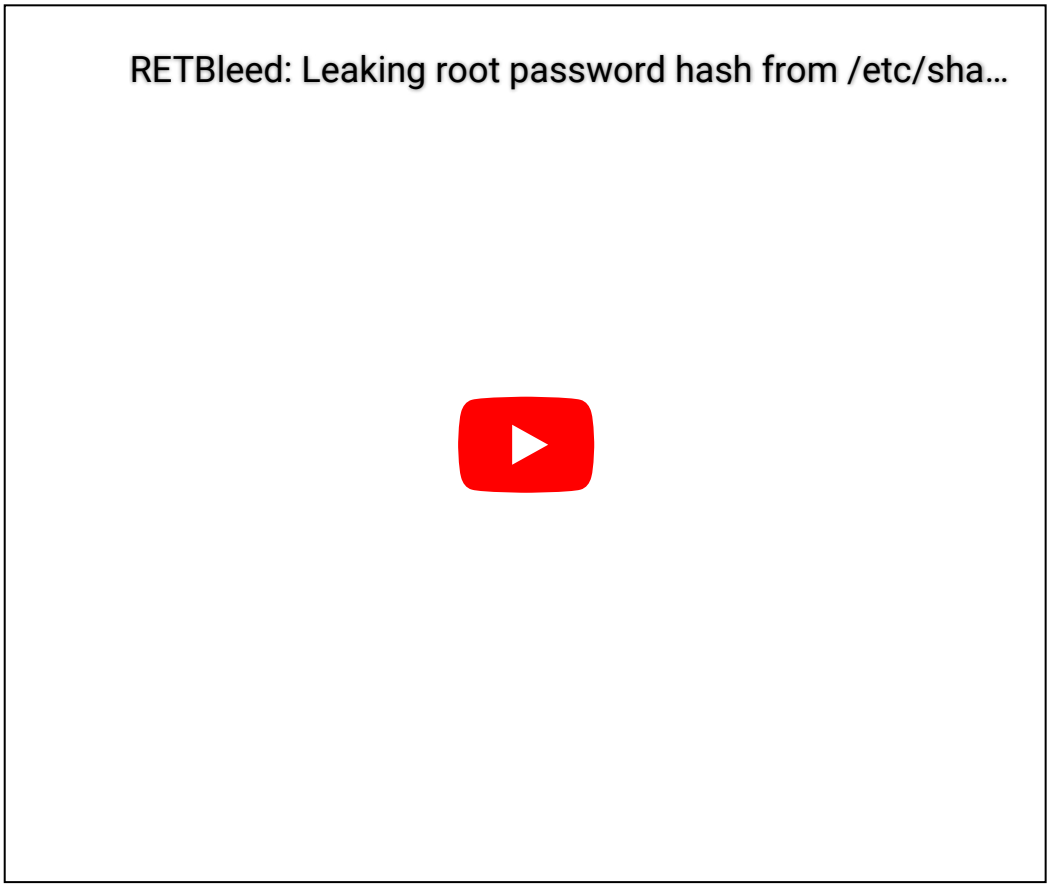
## Mitigations

Kernel and hypervisor developers have developed mitigations in coordination with Intel and AMD. Mitigating Retbleed in the Linux kernel required a substantial effort, involving changes to 68 files, 1783 new lines and 387 removed lines. Our

performance evaluation shows that mitigating Retbleed has unfortunately turned out to be expensive: we have measured between 14% and 39% overhead with the AMD and Intel patches respectively. Please refer to the paper if you want to know more. Mitigating Phantom JMPs with a generic flushing of the branch predictor unit on kernel transitions imposes up to 209% performance overhead.

## Demo

Below you can find a demo of Retbleed leaking kernel memory on Intel and AMD CPUs.




## Resources

A paper about Retbleed has been published and is going to be presented at USENIX Security 2022. You can find the source

code of Retbleed on our Github.

## FAQ

- **What's an indirect call/jump?**

  Indirect calls and jumps are branches where the branch target is determined at runtime. They are typically found in object-oriented languages to support polymorphism (e.g., objects inheriting from the same parent class), but also used in languages where function references can be passed as arguments (e.g., callbacks or lambda functions).

- **Why even predict returns like an indirect branch?**

  Return instructions are essentially indirect jumps fused with a pop from the stack. Because they are similar it makes sense to treat them as such from the microarchitecture's perspective.

- **Is my machine affected?**

  If it sports an AMD CPU family 0x15–0x17 or an Intel Core generation 6–8, it is likely affected. Note that we have only tested AMD CPU family 0x17 (AMD Zen 1, Zen 1+ and Zen 2).

- **Are these devices not from before Spectre and Meltdown were known?**

  Not all. In fact, AMD had not launched Zen2 when Spectre was originally published.

- **Are only Linux systems affected?**

  We've built the proof of concept code for Linux. But, because the fundamental issue is at the hardware level, Microsoft and Apple computers with the affected hardware have this issue too.

- **What should I do?**

  Install the latest operating system updates.

- **Should I worry?**

  If you have secrets on virtual machines with shared hardware (e.g., in the cloud), you should be aware of the issue. But it's not good for your health to worry too much.

©2022 Computer Security Group | Theme by SuperbThemes.Com