گزارش اعلام آسیب‌پذیری

عنوان گزارش: آسیب‌پذیری‌های Meltdown و Spectre

مرکز ماهر

بسمه‌تعالی

آسیب‌پذیری‌های Meltdown و Spectre

۱ مقدمه

چهارشنبه، مورخ ۳ ژانویه ۲۰۱۸ میلادی، محققان، مجموعه‌ای از آسیب‌پذیری‌های امنیتی اساسی را در ریزپردازنده‌ها (که حدود ۱۵ الی ۲۰ سال است که قلب تپنده‌ی کامپیوترها به‌شمار می‌روند) معرفی کرده‌اند. آسیب‌پذیری‌های مذکور، Meltdown و Spectre نام دارند. هردوی این آسیب‌پذیری‌ها، از طریق تنظیم مجدد ترتیب فرامین و یا اجرای فرامین مختلف به‌صورت موازی، مبادرت به دست‌کاری کارایی بهینه‌ی پردازنده‌ها می‌نمایند. مهاجمی که فرایندی را روی یک سیستم، کنترل می‌کند می‌تواند از آسیب‌پذیری‌های مذکور جهت سرقت اطلاعات محرمانه‌ی موجود در هر نقطه از کامپیوتر استفاده نماید.

۲ توصیف آسیب‌پذیری

تمامی آسیب‌پذیری‌های اعلام‌شده‌ی اخیر، از امتیاز مکانیزم اجرای احتمالی[۱] CPU بهره می‌جویند. در نگاهی ساده، CPU، یک ماشین قطعی بوده که در حال اجرای فرامین، به‌ترتیبی از پیش تعیین‌شده است. CPUهای دنیای واقعی، پیچیده‌تر هستند و این پیچیدگی، دری را به‌روی حملات ناخوشایند گشوده است.

معمولا یک CPU، بنابر ملاحظات کارآیی، بر اساس اجرای فرامین متعدد در یک زمان کار می‌کند. اجرای موازی فرامین، به پردازنده اجازه می‌دهد تا تعداد بیشتری از زیرمجموعه‌های خود را در یک زمان، مشغول نگه دارد. بنابراین، سرعت افزایش می‌یابد؛ اما، اجرای موازی منجر به کاهش سرعت دسترسی به حافظه‌ی اصلی نیز می‌شود. حافظه‌ی نهانی (کَشی)[۲] که نیاز به واکشی از RAM را از دست داده‌است می‌تواند اجرای یک دستور را برای صدها چرخه‌ی پردازنده، متوقف سازد. در نتیجه، تاثیر شایانی بر کارایی خواهدداشت.

CPU می‌تواند برای کمینه نمودن مقدار زمانی که در انتظار داده به‌سر می‌برد، فرامین را پس از نمونه‌ی متوقف‌شده، اجرا نماید. اساسا این روش، منجر به مرتب‌سازی مجدد کد در برنامه می‌شود. این مرتب‌سازی

[۱] Speculative execution

[۲] Cache

مجدد، غالبا نامشهود است، اما گاهی سبب می‌گردد تا سند Documentation/memory-barries.txt (راهنمایی جهت استفاده از موانع مختلف حافظه، که برای لینوکس تعبیه شده‌است) نوشته شود.

اجرای خارج از نوبت، چالش‌هایی را به هنگام انشعاب یافتن کد (دستورالعملی که به کد اعلام می‌دارد تا به بخش دیگری پرش کند)، به‌همراه خواهدداشت. ممکن است پردازنده هنوز قادر نباشد بگوید که کدام انشعاب (پرش) صورت پذیرد. بنابراین، نمی‌داند به کجا برود تا فرمان (فرامین) متوقف‌شده را از سر گیرد. پاسخ در این‌جا، پیش‌گویی انشعاب[۳] است.

پردازنده، بر اساس تجربه‌ی گذشته در ارتباط با انشعاب مورد نظر، و احتمالا، راهنمایی صریح برگرفته از کد (به‌عنوان مثال، دستور unlikely () به‌کار رفته در کد هسته‌ی سیستم‌عامل (کِرنِل)[۴]، گمانه‌زنی خواهدکرد. به‌محض آن‌که شرایط انشعاب ارزیابی گردد، پردازنده تشخیص خواهدداد که آیا حدسی که زده‌است صحیح بوده یا خیر. اگر صحیح نباشد، فرامین مبتنی بر گمان اجراشده‌ی پس از انشعاب، ملغی خواهدشد و همه‌چیز به حالت اول باز خواهدگشت؛ گویی که هرگز اجرا نشده‌اند.

در حقیقت، شکست در پیشگویی انشعاب، اجرای کُندتر و اثرات جانبی نهان را به‌همراه خواهدداشت. این امر، منجر به آسیب‌پذیری‌های مرتبط با افشای اطلاعات خواهدشد. مشخصا، اجرای فرامین مبتنی بر گمانه‌زنی می‌تواند منجر شود تا داده، روی حافظه‌ی نهان حافظه‌ی CPU بارگذاری گردد. سپس، زمینه برای استفاده از حملات زمان‌بندی فراهم می‌شود تا بیاموزند کدام فرامین اجرا شدند. در صورتی‌که اجرای گمانه‌زنی کد کرنل بتواند توسط یک مهاجم کنترل گردد، مضامین حافظه‌ی نهان قادر خواهندبود به‌عنوان یک کانال پنهان، جهت دست‌یابی به داده‌های خارج از کرنل به‌کار روند.

۱,۲ بررسی‌های مرزی

شاید زشت‌ترینِ آسیب‌پذیری‌ها، از نقطه‌نظر هزینه‌ی دفاع در برابر آنها، امکان پیش‌دستی در بررسی‌های مرز طبیعی[۵] را در کرنل فراهم آورد. فرض کنید کد کرنل مشابه زیر است:

---

[۳] Branch prediction

[۴] Kernel

[۵] Native Bound

```
if (offset < array1->length)
{
	unsigned char value = array1->data[offset];
	unsigned long index = ((value&1)*0x100)+0x200;
	if (index < array2->length) // length is < 0x300
	unsigned char value2 = array2->data[index];
}
```

اگر *offset* بزرگتر از طول *array1* باشد، هرگز نباید ارجاع به *array1->data* رخ دهد. اما اگر *array1->length* کَش نشود، پردازنده، در حین تست، متوقف خواهدشد. شاید، در حالی‌که در انتظار به‌سر می‌برد، پیش‌بینی کند که *offset* در محدوده قرار دارد (زیرا غالبا این‌گونه است) و اجرا را تا حدی به پیش ببرد که حداقل، واکشی مقدار *array2* را آغاز نماید. زمانی‌که *offset* خیلی بزرگ باشد، تمامی کارهای مرتبط با گمانه‌زنی‌های انجام‌شده، لغو خواهندشد.

در شرایطی‌که *array2->data[index]* در حافظه‌ی نهان حضور یابد، یک اکسپلویت قادر خواهدبود داده‌ها را در هردوی 0x200 و 0x300 واکشی کرده و زمان‌بندی‌ها را مقایسه کند. اگر یکی بسیار سریع‌تر از دیگری باشد، نمونه‌ی سریع‌تر، کَش می‌شود؛ این بدان معنا است که انشعاب داخلی، از طریق گمانه‌زنی انجام گردیده و مشخصا، طی این فرایند، کم‌ارزش‌ترین بیت *value*، تنظیم نشده‌است. فرایند مذکور، یک بیت از حافظه‌ی کرنل را، تحت کنترل مهاجم، نشر می‌دهد. البته، به‌کارگیری یک روش ماهرانه‌تر می‌تواند منجر به نشت بیش از یک بیت (کم‌ارزش‌ترین بیت) شود.

اگر الگوی کدی، مشابه الگوی فوق، در کرنل وجود داشته‌باشد و *offset*، در اختیار فضای کاربر قرار گیرد، این نوع حمله می‌تواند جهت نشت داده‌های دلخواه از کرنل به فضای کاربری مهاجم به‌کار رود. به‌نظر می‌رسد چنین الگویی وجود دارد و می‌تواند منجر به خوانده شدن داده‌ها از بیرون کرنل گردد. همچنین، این امکان وجود دارد که الگوی مورد نیاز، توسط برنامه‌ی BPF ایجاد شود- برخی از انواعی که می‌توانند بدون در اختیار داشتن هیچ‌گونه مزیتی، بارگذاری و اجرا گردند. حمله، جهت اجرا، نیرنگ‌آمیز است؛ به آماده‌سازی دقیق حافظه‌ی نهان CPU نیاز دارد؛ و به پردازنده وابسته است. اما، می‌تواند انجام پذیرد. پردازه‌های Intel، AMD، و ARM نسیت به این حمله، آسیب‌پذیر هستند. البته، درجه‌ی آسیب‌پذیری آنها متفاوت است.

راه صریحی جهت مقابله با این حمله وجود ندارد. به‌نظر می‌رسد، تنها تکنیک شناخته‌شده، ممانعت از اجرای احتمالی کدها داخل انشعابات، در زمانی است که شرایط انشعاب، تحت اختیار مهاجم قرار دارد. در این روش نیاز است تا پس از هر تست، مانعی قرار داده‌شود (که به نوبه‌ی خود، آسیب‌پذیر است). تعدادی وصله اولیه اعلام شده‌اند تا یک API جدید را برای ارجاعات اشاره‌گر حساس، اضافه کنند:

```
value = nospec_load(pointer, lower, upper);
```

این ماکرو، مقدار اشاره‌شده توسط *pointer* را بازخواهد گرداند، اما تنها در محدوده‌ی *lower* و *upper* موجود، قرار خواهدگرفت؛ در غیراین‌صورت، مقدار ۰ بازگردانده خواهدشد. شماری از انواع، در این ماکرو وجود دارند. روش مذکور، به دو دلیل مبهم است: نخست این‌که، به کارایی ضربه وارد می‌کند؛ از سویی دیگر، لازم است شخصی، الگوهای آسیب‌پذیر را در وهله‌ی اول بیابد. این امکان وجود دارد که آسیب‌پذیری‌های کنونی برطرف شوند، اما شکی نیست که آسیب‌پذیری‌های جدید این‌چنینی، به‌طور منظم معرفی خواهندشد.

۲,۲ آلوده نمودن از طریق پرش‌های غیرمستقیم[۶]

کرنل، به کرّات، از پرش‌های غیرمستقیم (به‌عنوان مثال، فراخوانی یک تابع از طریق یک اشاره‌گر) استفاده می‌کند. پیش‌گویی انشعاب متعلق به پرش‌های غیرمستقیم، از نتایج کَش‌شده در یک بافر مجزا، که تنها به ۳۱ بیت از آدرس موردنظر، کلید تخصیص می‌دهد، استفاده می‌نماید. اسم مستعار حاصل می‌تواند جهت مسموم کردن این مشکل به‌کار رود و سبب گردد که اجرای احتمالی، به مکان اشتباهی پرش کند. مجدداً، CPU متوجه می‌شود که در اشتباه به‌سر می‌برد و نتایج پرش بد را ملغی می‌نماید، اما اجرای احتمالی مذکور، اثرات را در حافظه‌ی نهان حافظه، رها خواهدکرد. این معضل می‌تواند اکسپلویت شده و منجر به اجرای احتمالی کد دلخواهی، که مجدداً امکان فیلتر نمودن داده‌ها را از کرنل فراهم خواهدکرد، شود.

یک جنبه‌ی ترسناک دیگر آسیب‌پذیری مذکور، این است که مهاجمی که در حال اجرا در یک میهمان مجازی‌شده به‌سر می‌برد، می‌تواند از آن برای نشت داده‌های قابل دست‌یابی به هایپروایزر[۷] (لایه‌ی نرم‌افزاری برای ایجاد محیط مجازی)، و یا به‌عبارتی دیگر، تمام داده‌های موجود در سیستم میزبان، استفاده کند. این امر،

[۶] Indirect Jump

[۷] Hypervisor

تمامی پیامدهای ناگوار را برای تامین‌کنندگان سیستم ابری، به‌همراه خواهدداشت. تنها، باید خوش‌شانس باشند که سیستم‌های خود را به‌روزرسانی کرده‌باشند.

دو راه برای مقابله با این معضل وجود دارد: راه نخست، به‌روزرسانی میکروکد متعلق به Intel است که این مشکل را حداقل، برای برخی از پردازنده‌ها برطرف می‌سازد؛ در غیاب این به‌روزرسانی، باید فراخوانی‌های غیرمستقیم، با یک ترمپولین[۸] (ترکیب کد مخصوصی به نام retpoline که می‌تواند با به‌کارگیری کامپایلرهای جدید، فراخوانی‌های غیرمستقیم را بدون گمانه‌زنی، انجام دهد) دو مرحله‌ای جایگزین شوند. این روش، از اجراهای احتمالی بیشتر، ممانعت به‌عمل خواهدآورد.

هزینه‌ی کارایی ترمپولین، قابل توجه خواهدبود. مجموعه‌ای از وصله‌های GCC، به‌زودی به بازار عرضه خواهندشد تا یک پرچم را (-mindirect-branch=thunk-exter) اضافه کنند و بدین‌ترتیب، ترمپولین‌ها را در صورت لزوم، به‌صورت خودکار تولید نمایند.

۳,۲ مجبور ساختن بارگذاری مستقیم حافظه‌ی نهان

آسیب‌پذیری نهایی، کاملا در فضای کاربر اجرا می‌شود؛ بدون این‌که کرنل را دربرگیرد. نوعی از کد فوق را تصور کنید:

```
if (slow_condition)
{
	unsigned char value = kernel_data[offset];
	unsigned long index = ((value&1)*0x100)+0x200;
	if (index < length)
	unsigned char value2 = array[index];
}
```

در این‌جا، *kernel_data*، یک اشاره‌گر فضای کرنل است که باید کاملا برای برنامه‌ی فضای کاربر، غیرقابل دست‌یابی باشد. ممکن است همان معضلات متعلق به اجرای احتمالی، باعث شوند تا بدنه‌ی بلوک *if* بیرونی (و در صورتی‌که بیت کم‌ارزش *value*، هویدا باشد، بلوک درونی) بر مبنای یک گمانه‌زنی اجرا گردد. یک مهاجم می‌تواند با بررسی زمان‌بندی‌های در دسترس، ارزش یک بیت از *kernel_data[offset]* را مشخص

[۸] Trampoline

کند. البته، مهاجم باید در وهله‌ی اول، یک اشاره‌گر کرنل مفید را پیدا نماید؛ اما، نوعی از این حمله می‌تواند جهت یافتن محل قرارگیری کرنل در حافظه‌ی مجازی به‌کار رود.

پاسخ به این حمله، ایزوله‌سازی صفحه‌ی جدول کرنل است، که داده‌ی فضای کرنل را کاملا برای فضای کاربر، غیرقابل رویت می‌سازد؛ به‌نحوی‌که نتواند در اجرای احتمالی، استفاده شود. این مورد، تنها یکی از سه معضلی است که توسط ایزوله‌سازی صفحه‌ی جدول برطرف می‌گردد. روش مذکور، به‌تنهایی، هزینه‌ی کارایی ۵ الی ۳۰ درصدی یا بیشتر را تحمیل خواهدکرد. به‌نظر می‌رسد پردازنده‌های Intel و ARM نسبت به این معضل، آسیب‌پذیر باشند. پردازنده‌ی ADM، در معرض این خطر قرار ندارد.

۳ آسیب‌پذیری‌ها و افشاگری‌های رایج (CVE) متناظر

همان گونه که ذکر شد با نگاهی ساده تر و در قالب یک مثال پردازنده‌های سوپراسکالر مدرن هنگام اجرای دستورات ماشین برای افزایش سرعت، دستورات را قبل از رسیدن به زمان اجراکمی زودتر در پایپ‌لاین اجرا قرار می‌دهند و بصورت همزمان اجرا می‌کنند. اینکار برای دستوراتی که در یک شاخه (if) قرار دارند هم انجام می‌شود حتی قبل از اینکه شرط داخل if اجرا شود. اگر شرط if برابر با False شد، محاسبات انجام شده دور ریخته می شود. نقطه ضعف Meltdown از همین مسئله استفاده کرده و یک timing-attack انجام می‌دهد. هدف نهایی حمله این هست که یک برنامه بطور غیر مجاز به بخش‌هایی از حافظه اصلی دسترسی پیدا کند که بطور معمول اجازه دسترسی به آن را ندارد. مثلاً یک برنامه با سطح دسترسی معمولی، به حافظه کرنل دسترسی پیدا کند. جایی که احتمالا پسوردها یا کلیدهای محرمانه در آن قرار دارد. دستورات داخل if طوری تنظیم می‌شوند که ابتدا یک دسترسی به حافظه به جای دلخواه از حافظه کرنل صورت بگیرد و یک بایت داده خوانده شود. این دستور اگر خارج if بود به محض اجرا منجر به crash می‌شد و ما هیچ وقت مقدار این بایت از حافظه را نمی توانیم رویت کنیم. ولی در عمل، پردازنده، چون هنوز به زمان اجرای واقعی نرسیده، این فرآیند را براحتی انجام می‌دهد و چون مطمئن است که اگر شرط if برابر با True شد و به اجرای واقعی رسیدیم قبل از اینکه محتوای حافظه را به ما بدهد crash رخ خواهد داد. اما جای حساس دستور بعدی است. این دستور (در واقع دو دستور ماشین است که من برای سادگی فرض می‌کنم یک دستور است!) هم باید طوری تنظیم شود که قبل از اینکه شرط if بررسی شود داخل پایپ‌لاین اجرا قرار بگیرد. در این دستور به دو محل مجاز از حافظه ممکن است دسترسی صورت بگیرد. اینکه کدام محل انتخاب شود بستگی به محتوای

بایتی دارد که در مرحله اول خوانده شده بود. نکته نهایی این است که دو محل مجازی که در مرحله قبل گفتم طوری تنظیم می‌شود که یکی از آنها در کش پردازنده موجود باشد و یکی از آنها موجود نباشد. از اینجا به بعد حدس بسیار ساده است. شرط if طوری تنظیم می‌شود که برنامه در عمل وارد if نشود و از کار نیفتد ولی از روی زمان اجرای دستورات پایپ‌لاین متوجه می‌شویم که دسترسی به حافظه صورت گرفت یا دسترسی به کش. به این ترتیب متوجه می‌شویم که بیت مورد نظر ما صفر بوده است یا یک. بیت به بیت داده‌ها خوانده می‌شود و اطلاعات غیر مجاز بدست می‌آید. اصولا ممکن است سامانه کامپیوتری دارای تراشه های آسیب پذیر که از اجرای احتمالاتی و پیش‌گویانه سوپر اکالری بهره جویی می نماید و امکان انشعاب غیرمستقیم دارد امکان افشای احرازهویت‌نشده‌ی اطلاعات را برای کاربری، که از دسترسی کاربری محلی برخوردار است فراهم سازد و در فاز بعد از طریق آنالیز کانال جانبی حافظه‌ی نهان داده، فراهم آورد. سیستم آسیب‌پذیری‌ها و افشاگری‌های رایج (CVE)[۹]، سه نوع زیر را مبتنی بر توضیحات فوق تعیین کرده و به آسیب‌پذیری‌های مورد مطالعه نسبت داده است، که همچنان تحت آنالیز قرار دارند:

- CVE-2017-5715
- CVE-2017-5753
- CVE-2017-5754

۴ انواع آسیب‌پذیری

آسیب‌پذیری‌ها، به‌گونه‌ای بر سیستم‌ها تاثیر می‌گذارند که داده‌های حساس را در حافظه، ایزوله می‌سازند. آنها به یک مهاجم اجازه می‌دهند تا حافظه‌ی ممتاز یک پردازنده را با بهره‌برداری از اجرای موازی پردازش‌ها، دست‌کاری کنند. همچنین، به وی اجازه می‌دهند تا با استفاده از کد JavaScript در حال اجرا در یک مرورگر، به حافظه در فرایندهای مهاجم، دست‌یابد. محتوای حافظه می‌تواند شامل اطلاعات فشرده‌شدن کلیدها، رمزهای عبور، کلیدهای رمزنگاری، داده‌های سایر سیستم‌های مجازی روی همان سیستم، و یا سایر اطلاعات ارزشمند باشد. اکنون، دو آسیب‌پذیری شناخته شده‌اند که Meltdown و Spectre نامیده می‌شوند.

۱,۴ Meltdown

امنیت سیستم‌های کامپیوتری، اساسا وابسته به ایزوله‌سازی حافظه است؛ به‌عنوان مثال، طیف‌های آدرسی کرنل، به‌عنوان غیرقابل دست‌یابی، برچسب زده می‌شوند و از دست‌یابی کاربر، در امان خواهندبود. Meltdown،

[۹] Common Vulnerabilities and Exposures (CVE)

از اثرات جانبی اجرای خارج از نوبت موجود در پردازنده‌های مدرن سوءاستفاده می‌کند تا مکان‌های دلخواه حافظه‌ی کرنل را، که شامل داده‌های شخصی و رمزهای عبور است، بخواند. اجرای خارج از نوبت، یک ویژگی کارایی غیرقابل حذف است و در طیف گسترده‌ای از پردازنده‌های مدرن وجود دارد. حمله، مستقل از سیستم‌عامل است و بر هیچ‌یک از آسیب‌پذیری‌های نرم‌افزاری، متکی نیست. Meltdown، تمامی مفروضات امنیتی اعطاشده توسط ایزوله‌سازی فضای آدرس، مانند محیط‌های مجازی‌سازی ناقص[۱۰] و نیز هر مکانیزم امنیتی ایجادشده بر این اساس، را نقض می‌کند.

Meltdown، یک حمله‌ی جدید است که امکان غلبه بر ایزوله‌سازی حافظه را به‌طور کامل، با تعبیه‌ی روشی ساده برای هر فرایند کاربر، فراهم می‌آورد تا کل حافظه‌ی کرنل ماشینی که آن را اجرا می‌کند (شامل تمامی حافظه‌ی فیزیکی نگاشت‌شده در ناحیه‌ی کرنل) را بخواند. Meltdown، هیچ آسیب‌پذیری نرم‌افزاری را اکسپلویت نمی‌کند؛ در عوض، اطلاعات کانال جانبی در دسترس در غالب پردازنده‌های مدرن، مانند ریزمعماری‌های مدرن Intel از سال ۲۰۱۰ و نیز CPUهای سایر فروشندگان، را اکسپلویت می‌نماید.

اصولا برای موفقیت حملات کانال جانبی به‌طور معمول، باید در مورد الگوریتم یا جزییات اجرایی سیستم اطلاعاتی در دسترس باشد تا حمله منجر به نشت اطلاعات محرمانه شود. با این حال در حمله Meltdown به دشمنی که می‌تواند کد را روی پردازنده‌های آسیب‌پذیر اجرا نماید، اجازه می‌دهد تا از کل فضای آدرس کرنل (شامل حافظه‌ی فیزیکی نگاشت‌شده) نسخه‌برداری کنند. سادگی و قدرت Meltdown، به اثرات جانبی ناشی از اجرای خارج از نوبت بازمی‌گردد. آسیب‌پذیری فوق، در زمره‌ی CVE-2017-5754 قرار می‌گیرد.

Spectre ۲,۴

پردازنده‌های مدرن، از پیش‌گویی انشعاب و اجرای احتمالی استفاده می‌کنند تا کارایی را افزایش دهند. حملات Spectre، حاوی فرمانی هستند تا عملیاتی را که نمی‌توانند در طول اجرای صحیح برنامه رخ دهند، به‌صورت گمانه‌زنی (که اطلاعات محرمانه را از طریق کانال جانبی، برای دشمن منتشر می‌نماید) اجرا کنند. علاوه بر محدودیت‌های ایزوله‌سازی فرایند، که به‌کارگیری کد اصلی با خود به‌همراه دارد، حملات Spectre می‌توانند جهت نقض فرایند سندباکس مرورگر، از طریق نصب آنها به‌وسیله‌ی کد JavaScript قابل حمل، استفاده گردد.

[۱۰] Paravirtualization

در مجموع، Spectre، حمله‌ی عمومی‌تری بوده که مبتنی بر مفاهیم مشابه Meltdown است و بر پردازنده‌های ARM و ADM (که Meltdown نمی‌تواند بر آنها اثر بگذارد) اثر می‌گذارد. به‌علاوه، این بدان معنا است که تعمیرات (رفع مشکل) و کارهای پیرامون مرتبط با Meltdown، از حملات Spectre در امان نخواهندبود. Spectre، دو بردار حمله‌ای جدا، به نام‌های CVE-2017-5715 و CVE-2017-5753 را پوشش می‌دهد.

۵ توصیف شرکت‌های نامدار از Meltdown و Spectre

شرکت Google، گزارشی را در راستای دو آسیب‌پذیری مذکور منتشر کرده‌است:

	
Spectre، ایزوله‌سازی میان برنامه‌های کاربردی مختلف را نقض می‌کند. این آسیب‌پذیری، به مهاجم اجازه می‌دهد تا به برنامه‌های عاری از خطا حقه بزند (که بهترین شیوه است) تا اطلاعات محرمانه‌ی خود را نشت دهند. در حقیقت، بررسی‌های ایمنی بهترین شیوه‌ی مذکور، زمینه‌ی حمله را افزایش می‌دهد و برنامه‌ها را بیشتر در معرض آسیب‌های Spectre قرار می‌دهد. استفاده و برطرف نمودن Spectre، سخت‌تر از Meltdown است.	Meltdown، بنیادی‌ترین ایزوله‌سازی میان برنامه‌ی کاربر و سیستم‌عامل را نقش می‌کند. این حمله، به برنامه اجازه می‌دهد تا به حافظه و مضامین محرمانه‌ی برنامه و سیستم‌عامل، دسترسی پیدا کنند. اگر کامپیوتر شما دارای یک پردازنده‌ی آسیب‌پذیر است و سیستم‌عامل وصله‌نشده را اجرا می‌نماید، داده‌های حساس آن در معرض نشت قرار خواهندگرفت. این امر، هم کامپیوترهای شخصی و هم سیستم‌های ابری را تحت تاثیر قرار می‌دهد.

شرکت Intel اظهار داشته‌است که اکسپلویت‌ها نمی‌توانند منجر به خرابی، تغییر، یا حذف داده‌ها شوند. اما در صورتی‌که مهاجم (هکر) بتواند به رمزهای عبور و کلیدهای رمزنگاری دست یابد، این قضیه، قابل بحث خواهدبود.

دو اکسپلویت مذکور، به سه نوع تقسیم می‌شوند. انواع ۱ و ۲، مرتبط با Spectre و نوع ۳، مرتبط با Meltdown است. Intel، نسبت به هر سه نوع، آسیب‌پذیر است:

- نوع ۱: محدوده‌های دورزدن بررسی[۱۱] (CVE-017-5753)
- نوع ۲: تزریق انشعاب هدف[۱۲] (CVE-2017-5715)
- نوع ۳: بارگذاری حافظه‌ی نهان داده‌ی قلابی[۱۳] (CVE-2017-5754)

شرکت ADM، به سه نوع مذکور پاسخ داده است. ماتریس پاسخ AMD در زیر مشاهده می‌شود:

نوع	عنوان اعطاشده توسط Google	جزئیات
۱	محدوده‌های دورزدن بررسی	توسط به‌روزرسانی‌های نرم‌افزاری/سیستم‌عاملی، حل شده‌است تا در دسترس فروشندگان و تولیدکنندگان قرار گیرد. تاثیر کارایی جزیی انتظار می‌رود.
۲	تزریق انشعاب هدف	تفاوت‌های معماری AMD، به معنای آن است که خطر بهره‌برداری این نوع، نزدیک به صفر است. آسیب‌پذیری نوع ۲، تاکنون در AMD مشاهده نشده‌است.
۳	بارگذاری کَش داده‌ی قلابی	آسیب‌پذیری AMD صفر به‌علت تفاوت‌های معماری AMD.

۶ قربانیان

اساسا، هر کسی که دارای یک کامپیوتر است، در زمره‌ی قربانیان این حملات قرار می‌گیرد. این کامپیوتر می‌تواند شامل دستگاه‌های محلی، مانند لپ‌تاپ و کامپیوتر رومیزی (PC)، گوشی‌ها و تبلت‌ها، و دستگاه‌های IoT باشد. همچنین، می‌تواند سِرورها و سرویس‌هایی را که از آنها بازدید می‌کنید (مانند سیستم‌های ابری، که مجازی‌سازی را پیشنهاد می‌دهند) دربرگیرد.

[۱۱] Bounds check bypass

[۱۲] Branch target injection

[۱۳] Rogue data cache load

از مسایل مطرح‌شده می‌توان برداشت نمود که برنامه‌های مخرب روی گوشی کاربر می‌توانند داده‌ها را از سایر برنامه‌های موجود در آن گوشی بدزدند؛ یا یک برنامه‌ی مخرب روی کامپیوتر کاربر (مانند مرورگر ویندوزی که وی توسط آن، در حال بازدید از یک وب‌سایت ناقص و یا اجرای بدافزاری که از یک حمله‌ی فیشینگ نشات می‌گیرد است) می‌تواند داده‌های موجود در هر نقطه از کامپیوتر او را به سرقت برد. سرویس‌های ابری، که غالبا ماشین‌ها را میان چندین مشتری به اشتراک می‌گذارند، بیش از همه در معرض آسیب‌پذیری‌های مذکور قرار دارند. از این‌روی، برنامه‌هایی که در حال اجرا بر زیرساخت‌های ابری هستند و نیز کاربران انتهایی برنامه‌های ابری، از قبیل Google Drive، به‌شدت تحت تاثیر این آسیب‌پذیری‌ها قرار می‌گیرند. هر شخصی می‌تواند یک فرایند را در فضای ابری ایجاد کند و داده را از هر کاربر دیگری که مشغول کار روی همان سخت‌افزار است، سرقت نماید.

سیستم‌هایی که در حال حاضر، بیش از همه در معرض این آسیب‌پذیری‌ها قرار دارند، مجموعه‌تراشه‌های Intel هستند، زیرا آسان‌ترین نوع جهت اکسپلویت شدن به‌شمار می‌روند؛ در نخستین مستندی که در این زمینه منتشر شده‌است، Intel مورد هدف قرار گرفته‌است. این بدان معنا است که در حال حاضر، لپ‌تاپ کاربر بیشتر از تلفن وی در معرض خطر قرار دارد.

۷ نحوه‌ی حمله

یک مهاجم باید برای اکسپلویت کردن آسیب‌پذیری‌های مذکور، کد را روی یک سیستم محلی اجرا کند. این امر می‌تواند به طرق گوناگون انجام پذیرد. ورود به‌صورت محلی، حتی به‌عنوان یک کاربر سطح پایین یا غیرممتاز، به مهاجم اجازه می‌دهد تا حمله را انجام دهد. همچنین، در صورتی‌که مهاجمان بتوانند کدهای مخرب را روی یک سیستم محلی اجرا کنند، قادر خواهندبود حمله را از راه دور انجام دهند. این امر می‌تواند در قالب بدافزار دانلودشده، بدافزار موجود در وب‌سایت‌های مخرب، و یا حتی از طریق اسناد مخرب ظاهر گردد.

۱,۷ بررسی سرکش بودن حمله[۱۴]

تاکنون، هیچ مستندی دال بر این‌که حمله‌ی مذکور، سرکش است، ارائه نگردیده‌است. به‌دلیل آن‌که محققان امنیتی، آسیب‌پذیری‌های مذکور را افشا کرده‌اند و این آسیب‌پذیری‌ها طی فرایند حمله‌ی فعال کشف نشده‌اند،

[۱۴] Wild Attack

محتمل است که مهاجمان، به اندازه‌ی سایرین، از آنها مطلع نباشند. اگرچه، این روند به‌سرعت تغییر خواهدکرد، زیرا اکسپلویت‌های اثبات مفهوم، پیش از این نوشته شده و حول اینترنت، انتشار یافته‌اند. احتمالا طی یک برهه‌ی زمانی، آسیب‌پذیری‌های فوق را به‌صورت اکسپلویت‌شده، در قالب بدافزارها و حملات محلی مشاهده خواهیم کرد.

۸ راه‌حل

به‌دلیل این‌که معضلات مطرح‌شده، مرتبط با سخت‌افزار هستند و به‌طور گسترده، بسته به نرم‌افزار مشخص، تغییر می‌یابند، راه‌حل‌های کامل، پیچیده خواهندبود و احتمالا، مدت زمانی را به خود اختصاص خواهندداد. خوشبختانه، همان‌گونه که اکسپلویت کردن Meltdown راحت‌تر است، برطرف نمودن آن نیز آسان‌تر خواهدبود. به‌همین ترتیب، همچنان‌که اکسپلویت کردن Spectre مشکل‌تر است، برطرف نمودن آن نیز از دشواری بیشتری برخوردار است.

فروشندگان سیستم‌عامل‌های کنونی، از قبیل Microsoft، Apple، و Linux در حال انتشار وصله‌هایی هستند که مکانیزم‌های حفاظتی را در برابر Meltdown تعبیه خواهند کرد. این وصله ها، با حذف نگاشت هسته‌ی به اشتراک‌گذاشته‌شده، که از قابلیت پیشگویی مقادیر در حافظه‌ی محافظت‌شده ممانعت به‌عمل می‌آورد، کار می‌کند. متاسفانه، با حذف این ویژگی، کارایی‌های پردازشی بسیاری نیز حذف می‌گردند که در نتیجه، منجر به کاهش کارایی سیستم‌های مذکور خواهدشد. تنزل کارایی، به میزان و نحوه‌ی اعتماد نرم‌افزار به دسترسی مذکور بستگی دارد. برآوردهای کنونی، از کاهش ۵ الی ۳۰ درصدی کارایی نرم‌افزار حکایت دارند.

در حال حاضر، حفظ یک PC ویندوز، بغرنج است و هنوز ناشناخته‌های بسیاری وجود دارد. Microsoft، Google، و Mozilla در حال صدور وصله‌هایی برای مرورگرهای خود (به‌عنوان نخستین گام دفاع) هستند. همان‌گونه که آخرین نسخه‌ی Internet Explorer (IE) و Edge، از یک نسخه‌ی تعمیر برای Windows10 برخوردارند، Firefox57 (آخرین نسخه) نیز شامل راه‌حلی برای نقص است. Google می‌گوید که راه‌حلی برای Chrom64 تعبیه نموده‌است که در ۲۳ ژانویه منتشر خواهدشد. Apple هنوز در مورد این‌که طرحی برای تعمیر مرورگر Safari یا حتی MACOS ارائه نداده است. کاربران Chrome، Edge، و Firefox باید به‌روزرسانی‌های خودکار را انجام دهند.

Microsoft، یک وصله امنیتی اضطراری را از طریق به‌روزرسانی ویندوز صادر کرده‌است، اما اگر در حال اجرای نرم‌افزار ضدویروس شخص ثالث باشید، احتمال دارد که وصله را مشاهده نکنید. محققان امنیتی در تلاش هستند تا لیستی از نرم‌افزارهای ضدویروس را که پوشش داده می‌شوند، تفسیر و ارائه نمایند.

اگر یک PC یا لپ‌تاپ مبتنی بر ویندوز دارید، بهتر است اطمینان حاصل نمایید که آخرین به‌روزرسانی‌های Windows10 و BIOS متعلق به Dell، HP، Lenovo، یا سایر سازندگان را در اختیار دارید. Microsoft یا Intel امیدوارند که ابزاری ساده (دارای یک اسکریپت PowerShell) را ایجاد کنند تا حفاظت از هردوی سفت‌افزار و به‌روزرسانی‌های ویندوز را بررسی نماید. اما، تا زمان پیدایش ابزار، لازم است بررسی‌های مذکور، به‌صورت دستی انجام پذیرند و یا کاربران با PowerShell آشنا شوند. در ادامه، فهرستی از بررسی مرحله-به-مرحله‌ی سریع مشاهده می‌گردد:

- اگر از یکی از مرورگرهای Chrome یا Firefox57 استفاده می‌کنید، آن را به آخرین نسخه، به‌روزرسانی نمایید.
- به‌روزرسانی ویندوز را بررسی کرده و از نصب بودن KB4056892 (وصله مختص Meltdown و Spectre)، اطمینان حاصل کنید.
- وب‌سایت OEM متعلق به PC خود را برای دریافت اطلاعات پشتیبانی و به‌روزرسانی سفت‌افزاری بررسی کنید و آنها را بلافاصله به‌کار بندید.

ضمنا، از ضمیمه‌های ایمیل، مستندات، و وب‌سایت‌های مشکوک بپرهیزید؛ از رمزهای عبور طولانی و پیچیده استفاده کنید تا از دسترسی کاربران احرازهویت‌نشده و نامعتبر به سیستم شما، ممانعت به‌عمل آید؛ و در انتها، نرم‌افزار خود را توسط وصله‌ها، به‌روز نگه دارید.

۹ مراجع

[1]. Wikipedia, “Meltdown (security vulnerability),” https://en.wikipedia.org/wiki/Meltdown_(security_vulnerability).

[2]. “Spectre and Meltdown Attacks Against Microprocessors,” https://www.schneier.com/blog/archives/2018/01/spectre_and_mel_1.html.

[3]. T. Warren, “How to protect your PC against the major ‘Meltdown’ CPU security flaw,” Jan 2018, https://www.theverge.com/2018/1/4/16848976/how-to-protect-windows-pc-meltdown-security-flaw.

[4]. K. Sigler, “Overview of Meltdown and Spectre,” Jan 2018, https://www.trustwave.com/Resources/SpiderLabs-Blog/Overview-of-Meltdown-and-Spectre/.

[5]. LWN, “Notes from the Intelpocalypse,” https://lwn.net/SubscriberLink/742702/83606d2d267c0193/.

[6]. P. Alcorn, “Understanding The Meltdown And Spectre Exploits: Intel, AMD, ARM, And Nvidia,” Jan 2018, http://www.tomshardware.com/news/meltdown-spectre-exploits-intel-amd-arm-nvidia,36219.html.

[7]. “Vulnerability of Speculative Processors to Cache Timing Side-Channel Mechanism,” Jan 2018, https://developer.arm.com/support/security-update.

[8]. R. Grisenthwaite, “Whitepaper: Cache Speculation Side-channels,” Arm, Jan 2018.

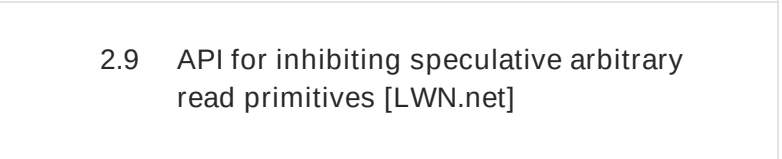
[9]. M. Lipp and et.al, “Meltdown,” https://meltdownattack.com/meltdown.pdf.

[10]. P. Kocher and et.al, “Spectre Attacks: Exploiting Speculative Execution,” https://spectreattack.com/spectre.pdf.

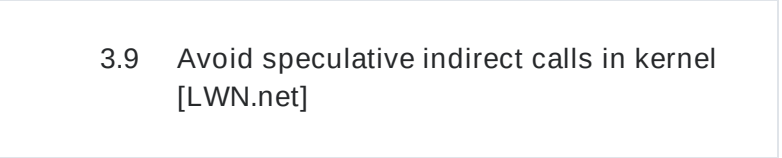
- لینک های بروزرسانی وصله های امنیتی

[January 3, 2018—KB4056892 (OS Build 16299.192)]

[API for inhibiting speculative arbitrary read primitives [LWN.net]]



[Avoid speculative indirect calls in kernel [LWN.net]]



[Dell Client Statement on Intel ME/TXE Advisory (INTEL-SA-00086) | Dell US]

The current state of kernel page-table isolation

5.9 The current state of kernel page-table isolation

At the end of October, the KAISER patch set was unveiled; this work separates the page tables used by the kernel...

kernel/git/torvalds/linux.git - Linux kernel source tree

6.9 kernel/git/torvalds/linux.git - Linux kernel source tree

arm64: Unmap the kernel whilst running in userspace (KAISER) [LWN.net]

7.9 arm64: Unmap the kernel whilst running in userspace (KAISER) [LWN.net]

[patch V163 00/51] x86/pti: Updated patch queue

8.9 [patch V163 00/51] x86/pti: Updated patch queue

Email message ("[patch V163 00/51] x86/pti: Updated patch queue") from Thomas Gleixner