Securité Informatique: Les Vulnérabilités Spectre et Meltdown

Arnold Kokoroko
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, Qc
arnold.kokoroko@umontreal.ca

Abstrait

Spectre et Meltdown sont des vulnérabilités critiques qui exploitent les failles des processeurs modernes en permettant à des programmes de voler des données qui sont actuellement traitées par des processeurs. Bien qu'un programme n'est généralement pas autorisé à lire les données d'autres programmes, en exploitant Spectre et Meltdown, un programme malveillant peut s'emparer de secrets conservés dans la mémoire d'autres applications en cours d'exécution. Il peut s'agir de vos mots de passe entreposés dans un gestionnaire de mots de passe ou un navigateur, de vos photos personnelles, de vos e-mails, de vos messages instantanés et même de documents critiques pour l'entreprise. Meltdown et Spectre fonctionnent sur les ordinateurs personnels, les appareils mobiles et dans le nuage. Dépendemment de l'infrastructure du fournisseur, dans le pire des cas, il serait même possible de voler les données d'autres clients dans le nuage.

1. Introduction

De nos jours, lorsque nous parlons de sécurité informatique, d'importants sujets qui nous viennent à l'esprit comprennent l'ingénierie sociale, la sécurité des modes d'authentification et des mots de passe, la sécurité de SHA et de RSA, etc. Cependant, que se passerait-il si nous faisions face à une attaque qui ne nécessite ni l'ingénierie sociale, ni de voler un mot de passe, ni d'utiliser une technique de décryptage? Une attaque nous permettant de lire les données sauvegardées dans un navigateur Web, d'espionner l'activité d'autres programmes ou ultimement d'accéder à toute l'information sauvegardée sur l'ordinateur d'une victime? Dans tous les cas, il s'agirait d'une situation très alarmante. Particulièrement, si cette situation était rendue possible sans l'utilisation de vulnérabilités logicielles. Il s'agit étonnamment de l'état actuel des choses qu'une équipe de chercheurs de Google Project Zero ainsi que de plusieurs universités ont publié en janvier 2018 décrivant une nouvelle attaque permettant à pratiquement n'importe quel programme de lire des données normalement inaccessibles [1]. Il s'agit des vulnérabilités Meltdown et Spectre [2][3].

Ces vulnérabilités au niveau du matériel, ou $hardware$, permettent des opérations aussi simples que lire une valeur après la fin d'un tableau ou $array$—une telle opération peut donner accès à des données qui ne devraient pas être accessibles dans le programme—jusqu'à, dans le pire des cas, être capable de lire dans la mémoire de l'espace noyau, ou $kernel$, qui est la mémoire du système d'exploitation où sont disposés les mots de passe et les clés de cryptage. En d'autres mots, avoir la capacité d'accéder à la mémoire du $kernel$ revient à avoir accès à toute la mémoire d'un appareil [4].

Tel que mentionné plus haut, il ne s'agit pas d'un problème au niveau du logiciel, mais plutôt d'un problème dans la façon dont les processeurs modernes sont implémentés, ce qui empire les choses. Que ce soit AMD, Intel, ARM ou autres, tous leurs processeurs pourraient être touchés. Ce qui signifie que non seulement les ordinateurs sont affectés, mais aussi les téléphones intelligents, serveurs et tout appareil doté d'un processeur. D'autant plus que ces vulnérabilités ne peuvent pas être détectées par un antivirus. Puis pour rendre les choses encore plus compliquées, ces failles découvertes il n'y a qu'un peu plus d'un an, sont si fondamentales pour la conception des processeurs

que la plupart des processeurs fabriqués au cours des 20 dernières années sont vulnérables. Même les nouveaux prototypes à venir ne sont pas totalement sécurisés. Selon le Computer Emergency Response Team (CERT), une équipe d'intervention en cas d'urgence informatique des États-Unis, la solution la plus complète pour éliminer ces vulnérabilités serait de remplacer tous les processeurs dans les appareils ainsi que de repenser leur conception et architecture [5].

Pour discuter de cette série d'attaques, ce rapport présentera d'abord, d'une manière générale, trois éléments essentiels à la compréhension de ces vulnérabilités soit; le fonctionnement de la mémoire dans un système d'exploitation; l'exécution spéculative; ainsi que les attaques par canal auxiliaire ou *side-channel attacks* pour ensuite utiliser ces concepts pour détailler les vulnérabilités Spectre et Meltdown.

2. Contexte

2.1. La mémoire

Prenons une vue simplifiée d'un ordinateur, nous avons le processeur central (CPU), des entrées et sorties telles que le moniteur, le clavier ou la souris ainsi qu'un module de mémoire, tels qu'indiqué sur la Figure 1. Lorsque nous exécutons un programme sur notre ordinateur, le CPU charge un programme à partir d'un fichier sur le disque et l'exécute dans la mémoire en enchaînant une série de calculs intermédiaires [4]. Ces calculs sont exécutés et conservés dans la mémoire vive ou *random-access memory* (RAM). Cependant, la mémoire vive est très lente comparée à la vitesse de calcul du CPU. C'est pour cela que nous avons aussi la mémoire cache.

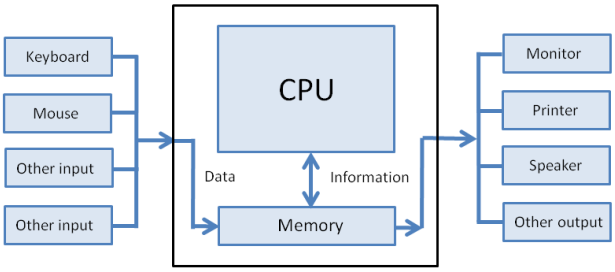

Figure 1. Architecture du matériel d'un ordinateur (Wikipédia)

La mémoire cache du CPU est une mémoire qui sert d'espace de rangement temporaire entre la mémoire vive et le CPU. Celui-ci peut accéder beaucoup plus rapidement à la mémoire cache tout en utilisant moins de cycles. Cependant, l'espace d'entreposage de la mémoire cache est limité comparé à la mémoire vive. Lorsque le CPU exécute un programme, il tente généralement de récupérer les données du programme et de les entreposer dans la cache. Une fois l'exécution terminée, il libère la cache pour créer de l'espace de rangement pour les autres programmes [4].

De cette façon, un CPU moderne est optimisé de façon à ce qu'il ne se sert pratiquement que de la mémoire cache pour maximiser son efficacité. Ceci est le concept à la base de l'exécution spéculative dont Meltdown et Spectre prennent largement avantage. Enfin, nous avons la protection de la mémoire qui est une protection intégrée dans le système d'exploitation empêchant un programme quelconque d'accéder à la mémoire d'autres programmes ou à une mémoire qui ne lui a pas été allouée [6]. C'est ce qui empêche, par exemple, une simple publicité sur le Web d'analyser tous les paquets passant par votre ordinateur ou de voler votre mot de passe Wi-Fi.

2.2. L'exécution spéculative

L'une des techniques que l'on retrouve sur presque tous les processeurs modernes est appelée l'exécution spéculative. C'est le moyen clé qui permet au CPU d'augmenter significativement ses performances. Pour parler de l'exécution spéculative, il faut tout d'abord d'écrire l'exécution dans le désordre ou *out-of-order execution*. Brièvement, il est plus difficile d'augmenter la vitesse d'horloge d'un processeur que de réordonner ses tâches. En utilisant ce dernier, nous pouvons optimiser l'ordre des instructions de sorte qu'elles puissent être lancées en parallèle ou dans l'arrière-plan. C'est ce que l'exécution dans le désordre fait. Cette technique permet au CPU de réorganiser des opérations afin qu'elles soient exécutées le plus efficacement et rapidement possible, et ce, sans altérer le résultat du code [7]. En d'autres mots, l'utilisateur ne doit pas s'en apercevoir.

Cela nous mène à l'exécution spéculative. Essentiellement, celle-ci permet au CPU d'essayer de deviner une branche d'opérations et d'exécuter les prochaines opérations d'avance tel qu'indiqué sur la Figure 2. En d'autres mots, le CPU exécute du code même s'il n'a pas nécessairement besoin de le faire, mais dont les chances que ce code soit nécessaire par la suite sont fortes. Dans la mesure où les prédictions se révèlent être fausses, le processeur annule simplement le résultat et réexécute ensuite le code avec les bonnes instructions [8]. Cette simple opération peut sauver des centaines de nanosecondes par instructions. La prédiction se fait à l'aide d'une fonctionnalité incluse dans les processeurs nommée prédiction de branchement [9]. Grâce à l'exécution spéculative faite par le CPU, un sys-

tème moderne peut lire à l'avance une page d'information ou traiter des données dont il pense avoir besoin sous peu et les emmagasiner dans la mémoire cache. Pour que cela ne cause pas de problèmes au CPU, l'hypothèse de l'exécution spéculative est qu'elle ne doit laisser aucune trace. Si la prédiction échoue, celle-ci doit être annulée et rembobinée [8]. En revanche, cette hypothèse ne s'avère pas toujours être vrai et c'est de cette faille dont Meltdown et Spectre prennent largement avantage.

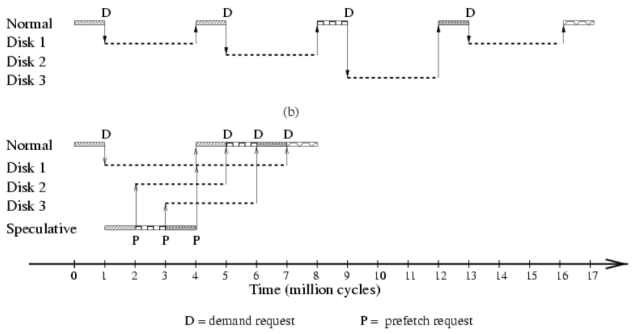

Figure 2. L'exécution normale d'une instruction en haut et l'exécution d'une instruction à l'aide de l'exécution spéculative en bas. L'instruction exécutée spéculativement est beaucoup plus rapide. (Usenix)

2.3. Les attaques par canaux auxiliaires

En sécurité informatique, la confidentialité, la conformité et l'accessibilité sont trois propriétés importantes largement appliquées à l'aide de SHA, RSA, des clés symétriques, TLS, etc. Cependant, diverses méthodes astucieuses ont été créées pour contourner ces propriétés. Certaines de ces méthodes prennent avantage de possibles fuites imprévues durant l'échange d'informations. Ces fuites—où d'importantes informations peuvent être divulguées—sont appelées canaux auxiliaires. Dans un ordinateur conventionnel, toute propriété mesurable de l'implémentation de l'ordinateur a le potentiel d'être un canal auxiliaire [8].

Une attaque populaire via un canal auxiliaire est l'attaque temporelle ou *timing attack*. Les attaques temporelles utilisent le temps en tant que propriété mesurable pour déduire des informations confidentielles en comparant le temps requis par diverses opérations. Un tel exemple est démontré dans la Figure 3 avec RSA. À cause des attaques temporelles, la confidentialité d'un système est largement menacée lorsque le code de différents programmes partage des ressources telles que les processeurs, la mémoire, la cache, ou l'espace sur le disque [8].

Prenons l'exemple de la cache d'un CPU dans un ordinateur

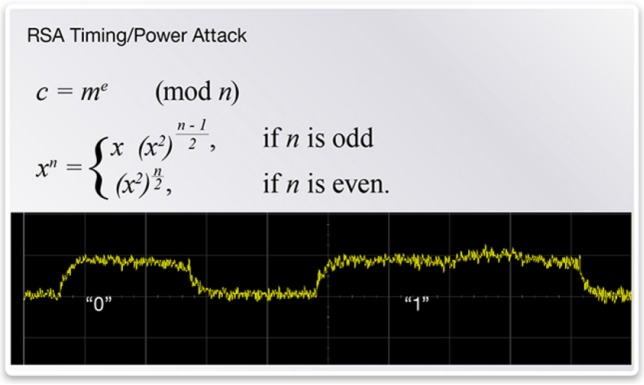

Figure 3. Exemple d'une attaque temporelle par canal auxiliaire sur RSA (Synopsys)

conventionnel. Celle-ci conserve l'adresse des emplacements récemment consultés dans la mémoire ainsi que leurs valeurs. Tel qu'expliqué plus tôt, la consultation d'une adresse dans la mémoire est très rapide pour les adresses mises en cache, ce qui permet de savoir quelles adresses ont été récemment consultées ou non. Un adversaire mal intentionné pourrait donc soutirer de l'information en exécutant du code de manière séquentielle ou concurrente de telle sorte que le CPU divulgue de l'information sur son contenu. C'est une attaque qui s'avère très efficace en cryptanalyse et qui permet de découvrir des informations sensibles sur les données mises en cache [8].

Pour démontrer de tels dangers, prenons un exemple fictif où un adversaire malicieux tente de s'authentifier sur un réseau passant par un routeur à partir d'un ordinateur et que le mot de passe est éventuellement conservé en clair dans la cache du routeur. En pratique, cela ne serait jamais le cas avec des barrières comme le salage et le cryptage par hachage. Maintenant supposons que le mot de passe est «Password123». Une méthode inefficace ici serait d'utiliser un dictionnaire contenant tous les mots de passe possibles ce qui nécessiterait un nombre exponentiel de possibilités. Une technique plus ingénieuse serait de deviner la première lettre du mot de passe en essayant disons «A». Le système vérifierait si la lettre «A» correspond au mot de passe en vérifiant le premier caractère du mot de passe et évidemment retournerait que celui-ci est invalide. Cette première lettre du mot de passe serait maintenant en cache. L'adversaire pourrait ensuite réessayer avec le caractère «B» et ainsi de suite en arrivant toujours au même résultat invalide jusqu'à ce qu'il essaye avec le caractère «P». Dans ce cas-ci, le système comparerait la lettre «P» au premier caractère du mot de passe, trouverait une correspondance et passerait ainsi au prochain caractère du mot de passe pour réaliser qu'il ne concorde pas et retourner invalide. Par conséquent, à cause de cette opéra-

tion additionnelle qui nécessite une valeur n'étant pas déjà dans la cache, le routeur prendrait ainsi une fraction de seconde de plus à répondre. Assez pour que l'adversaire puisse détecter cette différence de temps par rapport aux autres caractères et déduire que la première lettre du mot de passe est «P». En répétant ces étapes, l'adversaire pourrait déduire le reste du mot de passe en temps linéaire. En pratique, une telle attaque ne serait pas réalisable grâce à la sécurité additionnelle des systèmes informatiques, mais combinées avec d'autres types d'attaque, les attaques temporelles par canaux auxiliaires peuvent s'avérer très dangereuses.

3. Description des Vulnérabilités

3.1. Meltdown

De nos jours, une caractéristique au cœur de la sécurité des systèmes d'exploitation est la protection, ou isolation, de la mémoire. Les systèmes d'exploitation garantissent que les programmes ne peuvent pas accéder à la mémoire d'autres programmes ou à la mémoire dédiée au $kernel$. Cette isolation nous permet d'exécuter plusieurs applications en même temps sur des appareils personnels ou d'exécuter des processus venant de divers utilisateurs sur une machine dans le nuage [2]. Après avoir discuté des diverses propriétés des systèmes informatiques, nous présentons maintenant la vulnérabilité Meltdown qui permet de surmonter la propriété d'isolation de la mémoire.

Les vulnérabilités de Spectre et de Meltdown résident dans le fait qu'un processeur moderne puisse exécuter ses instructions aussi rapidement et efficacement que possible. Dans le cas de Meltdown, c'est une attaque qui permet de surmonter complètement l'isolement de la mémoire en fournissant un moyen simple pour tout processus utilisateur de lire la mémoire du $kernel$ d'une machine sur laquelle il s'exécute. Ce qui est aggravant est que Meltdown n'exploite aucune vulnérabilité logicielle. Cette attaque peut donc être utilisée sur tous les principaux systèmes d'exploitation. En fait, elle exploite les informations sur les canaux auxiliaires venant des processeurs modernes, principalement ceux d'Intel construits depuis 2010 [2]. Dans la situation décrite lors de l'introduction des attaques par canaux auxiliaires, celle-ci nécessitait des connaissances très spécifiques du système ciblé et elle ne finissait que par divulguer des fuites de renseignements qu'il fallait ensuite analyser pour réussir à découvrir l'information voulue. Cependant, dans le cas de Meltdown, un adversaire pouvant exécuter du code sur le processeur de la victime pourrait parvenir à obtenir un tronçon de l'espace d'adresse du $kernel$ y compris la mémoire protégée. Une telle attaque est en fait possible dû à l'exécution dans le désordre.

En réalité, deux points importants permettre à Meltdown d'utiliser des instructions exécutées dans le désordre pour lire dans la mémoire. Tout d'abord, les processeurs vulnérables à l'attaque permettre la lecture d'adresses non protégées par des instructions exécutées dans le désordre. Ensuite, ces processeurs contiennent une situation de compétition, ou *race condition*, entre la levée d'exceptions et l'exécution des instructions en désordre. L'attaque fonctionne de la manière suivante. Lorsqu'un programme d'utilisateur, exploitant la vulnérabilité Meltdown, commence par lire une valeur dans la mémoire du $kernel$, ce dernier soulève une exception car il n'est pas autorisé à lire ce contenu avec des privilèges d'utilisateur. Jusqu'ici tout fonctionne bien. Cependant, en raison de la situation de compétition, des instructions initialement exécutées dans le désordre peuvent également s'exécuter malgré l'exception soulevée. Ceci, même si les instructions apparaissent après que l'erreur soit générée par le système [5]. Donc en créant des instructions bien réfléchies, celles-ci peuvent contenir des données extraites de l'instruction qui a soulevé l'exception et contourner l'erreur. Donc, au moment où l'exception est soulevée, un certain nombre d'instructions dans le désordre sont exécutées et l'information du $kernel$ est soutirée. Bien que l'exception élevée entraîne le retour en arrière de toutes les instructions, l'état de la cache n'est en revanche pas réinitialisé permettant aux données provenant des instructions dans le désordre de persister au-delà du moment où l'exception a été soulevée.

Pour illustrer ce concept et l'étendre à deux programmes d'utilisateur A et B, disons que le programme A demande à voir des données dans la mémoire du programme B, le CPU lèvera une erreur, mais seulement après avoir commencé à lire dans la mémoire du programme B à cause de l'exécution dans le désordre. De ce fait, les données recueillies par les instructions de A se trouvent maintenant dans la cache du CPU, et bien que le CPU ne transmette pas directement ces données au programme A, celui-ci peut les déterminer en trompant le CPU: le programme A dans une instruction additionnelle avant que l'erreur ne soit propagée, demande au CPU d'ajouter la valeur maintenue en cache à un nombre arbitraire qui correspondra à une adresse dans la mémoire. Puis, à la suite de l'erreur, le processus A demande au CPU de lire plusieurs adresses sachant que celle qu'il cherche est aussi en cache. En notant celle qui se charge le plus rapidement à l'aide de l'analyse temporelle, le processus A peut ainsi trouver l'adresse qu'il recherche. Finalement, le processus A peut déterminer la donnée du processus B se trouvant dans l'adresse en cache puisque celle-ci ne sera que le nombre arbitraire choisi plus tôt plus la valeur de la donnée du programme B [2].

Meltdown défait tous les principes de sécurité fournis par les

capacités d'isolation de la mémoire du CPU. Cette attaque a été évaluée contre les ordinateurs de bureau, les portables, ainsi que les serveurs dans le nuage et Meltdown permet à un processus sans privilèges de lire les données entreposées dans la mémoire du $kernel$. Ceci inclut l'entièreté de la mémoire physique sous Linux, Android et MacOs, et une grande partie de la mémoire physique sous Windows. Tandis que la performance dépend fortement de la machine spécifique, celle-ci peut être copiée à une vitesse de 3.2 KB/s à 503 KB/s selon les chercheurs de Google Project Zero [2]. Les processeurs ciblés par Meltdown sont ceux d'Intel, car ils n'ont pas de contrôle de privilèges pour les instructions exécutées dans le désordre. Le contrôle pour s'assurer que l'instruction est valide n'est effectué qu'après avoir lu dans la mémoire. Cependant, pour les compagnies telles qu'AMD et ARM, le test de contrôle de privilèges est exécuté avant la lecture dans la mémoire. La vulnérabilité Meltdown ne les affecte donc pas [6].

3.2. Spectre

Si Meltdown ne touche principalement que les processeurs d'Intel, Spectre affecte tous les processeurs fabriqués au cours des deux dernières décennies, y compris ceux d'AMD, ARM et même d'IBM [10]. Spectre est un problème très semblable à Meltdown, mais qui s'appuie sur des techniques plus nuisibles et aussi plus difficiles à exploiter. Les attaques de Spectres consistent à effectuer des opérations spéculatives sur l'appareil d'une victime qui ne se produiraient pas durant le bon déroulement d'un programme. Ces attaques ont pour but de soutirer des informations confidentielles à la victime via un canal auxiliaire sans qu'elle ne s'en rende compte. Contrairement à Meltdown, qui peut être utilisé pour lire dans la mémoire du $kernel$, Spectre est utilisé pour manipuler un processus à révéler ses propres données.

Tel qu'introduit plus haut dans le document, l'exécution spéculative est une avancée qui a facilité l'augmentation de la vitesse des processeurs au cours des 20 dernières années. Celle-ci est largement utilisée pour augmenter les performances et implique que le CPU utilise un prédicteur de branche pour deviner les futures instructions et les exécuter prématurément. Une fois que les vraies données des instructions finissent par arriver de la mémoire vive, le CPU vérifie si son estimation était valide. Si l'estimation était fausse, le CPU rejette l'exécution spéculative et réinitialise les dernières instructions lancées. Si l'estimation était correcte, les résultats de l'exécution spéculative sont gardés et un énorme gain de performance est donc réalisé [3]. D'un point de vue de sécurité, l'exécution spéculative implique l'exécution d'un programme de manière potentiellement incorrecte. Cependant, comme les processeurs sont conçus pour retourner vers l'arrière et annuler l'exécution spéculative dans le cas où elle s'avère être incorrecte, ces opérations étaient—jusqu'à quelque peu—considérées sécuritaires et sans problème. En vrai, tous les effets secondaires de l'exécution spéculative ne sont pas annulés tels que les changements dans la cache du CPU. Donc, en effectuant une analyse temporelle des opérations accédant à la mémoire, la cache peut être utilisée pour extraire des informations secrètes venant d'instructions exécutées de manière spéculative [5].

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

Figure 4. Exemple simplifié d'un accès à la mémoire

Spectre possède deux variantes lui permettant d'attaquer un système. Dans la première, l'adversaire entraîne le prédicteur de branche du CPU à mal prédire la direction d'une branche. Ainsi, l'adversaire peut faire en sorte d'exécuter du code que le CPU n'aurait autrement pas exécuté [3]. Cette variante permet entre autres d'accéder à de la mémoire «hors limites». Prenons un exemple simplifié dans la Figure 4 qui simule l'accès à de la mémoire dans un ordinateur. Le tableau $array1$ simule l'espace réservé à un utilisateur et tout accès en dehors de ce tableau devrait être interdit par le système. Pour ce faire, la condition if s'assure que les accès au tableau array1 soient faits dans une plage légale délimitée par $array1\_size$. Normalement, grâce à cette condition, la variable x ne devrait jamais être en mesure de dépasser la taille limite du tableau $array1$. Cependant, une tactique qu'un adversaire malicieux pourrait employer pour accéder à des données secrètes en dehors des bornes du tableau serait tout d'abord de conditionner le prédicteur de branche du CPU à s'attendre à ce que la condition if soit évaluée vrai en invoquant le code avec des entrées valides de x en deçà de $array1\_size$. Après avoir répété cette étape pour plusieurs valeurs de x valides, l'adversaire invoque le code cette fois-ci avec une valeur invalide de x en dehors des bornes du tableau $array1$. À cause de l'exécution spéculative, le prédicteur de branche du CPU assume qu'il va recevoir une valeur valide et exécute l'instruction pour gagner du temps plutôt que d'attendre le résultat de la condition if. L'instruction $array2[array1[x] * 4096]$ est donc exécutée avec un x invalide et, si ingénieusement montée, peut accéder à de l'information secrète. Lors de cette exécution, l'instruction malicieuse est chargée dans la cache du CPU pour y avoir un accès plus rapide. Plus tard, lorsque le résultat de la condition if est finalement calculé, le CPU réalise son erreur et réinitialise les changements apportés. Cependant, les modifications chargées dans la cache ne sont pas annulées. L'adversaire peut ainsi analyser

le contenu de la cache à l'aide des canaux auxiliaires et obtenir l'information secrète [3].

Pour étendre cet exemple à une situation réelle, une personne pourrait surfer sur le Web avec plusieurs onglets actifs sur son navigateur et tomber sur le site d'un adversaire malicieux exécutant un code Javascript bien conçu. Ce code pourrait utiliser la vulnérabilité Spectre pour copier la mémoire du navigateur au-delà de celle qui lui est attribuée et ainsi obtenir de l'information secrète. Si la victime venait tout juste de visiter son compte bancaire et que la clé privée de celle-ci était chargée dans la cache pour effectuer les calculs de décryptage, l'attaquant pourrait ainsi subtiliser la clé privée et décrypter l'identifiant bancaire [6]. Heureusement, une telle attaque est difficile à orchestrer car elle nécessite une très haute connaissance des systèmes informatiques et une configuration parfaite, mais il reste que cette faille existe.

La deuxième variante est très similaire à la première où, au lieu d'accéder à de la mémoire au-delà de ses limites, l'adversaire amène le CPU à exécuter du code à une certaine adresse contenant de l'information importante. Les processeurs possèdent un historique de branchements ou *Branch Target Buffer* (BTB) leur permettant de garder des statistiques sur l'évaluation d'une condition if ainsi que l'emplacement de la prochaine instruction suivant la condition. Ici, le fonctionnement du *Branch Target Buffer* est simplifié, mais une explication plus exhaustive peut être trouvée dans le document de recherche sur Spectre publié par l'équipe de Google Project Zero [3]. Comme les processeurs n'ont pas une mémoire infinie, ils ne peuvent pas garder des statistiques sur tous les branchements qui existent. À cause de cela, l'historique de branchement ne range qu'une partie de l'adresse de l'instruction prédite au lieu d'utiliser une adresse complète [6]. Pendant l'exécution spéculative, lorsque le CPU cherche l'emplacement de la prochaine instruction à exécuter suite au branchement, une adresse relative est donc utilisée. Un attaquant peut ainsi conditionner le prédicteur de branche à pointer à une adresse cible prédéterminée et exécuter spéculativement des instructions à cette adresse [11]. Tout ce qu'il faut, c'est que l'espace de la victime soit à portée de l'adresse relative. L'adversaire pourra ensuite utiliser le même procédé que dans la première variante pour soutirer les informations dans la cache avec l'aide des canaux auxiliaires.

Face à Spectre, même la mémoire du $kernel$ n'est pas totalement sûre. Une attaque nommée «eBPF» a été découverte permettant indirectement à un programme d'utilisateur d'accéder à la mémoire du $kernel$ en lui injectant du code via Spectre [1]. Les machines virtuelles et le nuage ne sont pas non plus à l'abri. En théorie, une attaque utilisant les variantes de Spectre pourrait subtiliser n'importe quelle donnée sur un processeur. C'est pourquoi les machines virtuelles sur un ordinateur ou sur le nuage partageant le même processeur sont aussi vulnérables [10]. Cela va aussi de soi pour les différents systèmes multi-utilisateurs. Même les systèmes utilisés pour naviguer sur le Web peuvent être vulnérables. Seuls les systèmes mono-utilisateurs ne permettant pas facilement un accès local à un adversaire sont exposés à un risque nettement plus petit [5].

4. Comment se protéger?

Généralement, lorsqu'une vulnérabilité est découverte, celle-ci est rapidement corrigée lors d'un correctif ou $patch$ par les équipes de sécurité et de développement. Cependant, dans le cas de Spectre et Meldown, il semble que cela nécessitera plus qu'un simple correctif logiciel pour régler le problème. Pour le moment, certaines solutions ont été mises en place pour limiter les dégâts, mais selon les chercheurs de Google les vulnérabilités de Spectre et Meltdown devront nécessiter des changements complets au niveau du $hardware$ [12].

Le problème est que les processeurs modernes exécutent plusieurs programmes en utilisant le même $hardware$ sans isolation complète entre les programmes. En fait, ce manque d'isolement est intentionnel puisqu'il permet d'accélérer la vitesse d'exécution que ce soit à l'aide de la cache ou de *threads* qui communiquent entre eux. Ce manque d'isolement est donc difficile à éviter [12]. Tout juste après la découverte des vulnérabilités, des compagnies comme Intel, Microsoft, Linux, Apple, Google et Mozilla ont créé des $patchs$ pour adresser les vulnérabilités exploitant ce manque d'isolement. Par exemple, Mozilla a réduit la précision et désactivé plusieurs sources de temps pour contrer les attaques par canaux auxiliaires [13]. Google Chrome a développé une fonctionnalité appelée «Site Isolation» qui garantit que les pages de différents sites Web sont toujours placées en *sandbox* dans des processus différents [14]. Les compagnies Windows, Linux et Apple ont adopté une fonctionnalité créée spécialement pour contrer la vulnérabilité de Meltdown, nommé l'isolation de la table de pages du noyau ou *kernel page-table isolation* (KPTI). Le KPTI corrige les fuites des canaux auxiliaires en isolant plus efficacement la mémoire de l'espace d'usagers et de l'espace du $kernel$ en séparant leur table de pages entièrement [15]. Ces tables de pages sont des structures de données qui correspondent aux adresses virtuelles et physiques des espaces d'usagers et du $kernel$.

Ces différents $patchs$, dont l'implémentation du KPTI, pourraient cependant avoir de sérieuses répercussions sur

les performances des systèmes touchés. Selon certains chercheurs, ces pertes de performance pourraient aller jusqu'à 30% dans le pire des cas [16], alors que selon Intel, l'impact de l'implémentation du KPTI ne devrait pas être significatif pour un ordinateur ou appareil standard [17]. Bref, l'impact sur les performances risque de varier d'appareils en appareils dépendamment du processeur utilisé. Les plus vieux processeurs seront probablement les plus durement touchés.

En ce qui est d'Intel, AMD et des autres fabricants de processeurs, ceux-ci sont en train d'effectuer des modifications sur l'architecture de leurs nouveaux modèles. L'exécution spéculative est un fondement au cœur des processeurs pour augmenter leurs performances et résoudre toutes les failles générées par celle-ci s'avère être un problème complexe. De plus, Spectre est une vulnérabilité qui ne peut pas être entièrement résolue par une correction au niveau des logiciels. Bien qu'Intel ait déclaré avoir conçu des correctifs au niveau du $hardware$ pour ses futurs modèles adressant la vulnérabilité de Meltdown et la deuxième variante de Spectre, depuis ce temps plusieurs nouvelles variantes de Spectre et Meltdown ont été découvertes [18]. Jusqu'à aujourd'hui, ces vulnérabilités restent donc problématiques.

Pourtant, cela signifie que des milliards de processeurs en circulation sont donc vulnérables. Il va de soi que remplacer chacun d'entre eux est impossible. Pour l'utilisateur régulier, il y a de quoi s'inquiéter. Ceci étant dit, la meilleure façon de se protéger présentement face aux vulnérabilités de Spectre et Meltdown est de s'assurer de disposer des dernières mises à jour de sécurité sur tous ses appareils ainsi que des mises à jour de ses systèmes d'exploitation que ce soit Windows, Linux, Android ou MacOS. Heureusement, ces vulnérabilités restent encore difficiles à exploiter aujourd'hui, et s'assurer que son système est à jour et garder de bonnes habitudes sécuritaires en ligne peuvent continuer à nous tenir à part des problèmes pour le moment.

5. Conclusion

Les vulnérabilités Spectre et Meltdown ont eu un effet dévastateur sur la sécurité des processeurs en affectant une majorité des systèmes dans le monde. Dans le cas des virus ou des vers, ceux-ci affectent un système en exploitant les faiblesses dans le code d'un programme en particulier, ce qui signifie qu'ils ne peuvent pas affecter un système qui leur est inconnu. Cependant, Spectre et Meltdown touchent tous les processeurs utilisant l'exécution spéculative peu importe le système. Ils ne sont donc pas restreint quant à leur choix de victime. En fait, cela inclut presque tous les processeurs Intel fabriqués depuis 1995, la plupart des processeurs AMD, ainsi que les processeurs de tablettes et téléphones intelligents tels que l'iPhone et le Samsung. En utilisant l'exécution dans le désordre, l'exécution spéculative et la prédiction de branche, Meltdown et Spectre permettent à des programmes d'utilisateur sans privilège d'exécuter du code sur un système et de lire des informations sensibles depuis la mémoire.

Pour régler ce problème, diverses tentatives de correction ont été mises en place telles que les mises à jour des systèmes d'exploitation Windows, Linux et Apple ainsi que les mises à jour de différents navigateurs Web tels que Chrome et Mozilla. Cependant, la solution idéale pour éviter ces correctifs—jugés ad hoc—serait de redessiner l'architecture complète des processeurs modernes puis de les reconstruire avec l'aide d'ingénieurs et d'experts en sécurité informatique pour être certain que ces nouveaux prototypes répondront à de meilleurs objectifs de sécurité clairs et précis.

References

[1] O. Jann, "Reading privileged memory with a side-channel." Google Zero. https://googleprojectzero.blogspot.com/2018/01/reading-privileged-memory-with-side.html, Jan 2018. [Online; accessed 20-April-2019].

[2] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown," *CoRR*, vol. abs/1801.01207, 2018.

[3] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," *CoRR*, vol. abs/1801.01203, 2018.

[4] A. S. Tanenbaum and H. Bos, *Modern Operating Systems*. Upper Saddle River, NJ, USA: Prentice Hall Press, 4th ed., 2014.

[5] "Cpu hardware vulnerable to side-channel attacks." CERT Coordination Center. Software Engineering Institute. https://www.kb.cert.org/vuls/id/584653/, Jan 2018. [Online; accessed 20-April-2019].

[6] B. Hubert, "Spectre and meltdown: tapping into the cpu's subconscious thoughts." ds9a-nl articles. https://ds9a.nl/articles/posts/spectre-meltdown/, Jan 2018. [Online; accessed 20-April-2019].

[7] "Out-of-order execution." Wikipedia. https://simple.wikipedia.org/wiki/Out-of-order_execution, Jan 2018. [Online; accessed 20-April-2019].

[8] R. McIlroy, J. Sevcík, T. Tebbi, B. L. Titzer, and T. Verwaest, "Spectre is here to stay: An analysis of side-channels and speculative execution," *CoRR*, vol. abs/1902.05178, 2019.

[9] "Prédiction de branchement." Wikipedia. https://fr.wikipedia.org/wiki/Prédiction_de_branchement#Branch_target_buffer, Mar 2019. [Online; accessed 20-April-2019].

[10] A. Regidi, "Meltdown and spectre: The simple-english guide to the most worrying computer security flaw in decades." Firstpost. Tech2. https://www.firstpost.com/tech/news-analysis/meltdown-and-spectre-the-simple-english-guide-to-the-most-worrying-cpu-security-flaw-in-
html, Jun 2018. [Online; accessed 21-April-2019].

[11] "Cve-2017-5715 (spectre, variant 2) - cve." WikiChip. https://en.wikichip.org/wiki/cve/cve-2017-5715. [Online; accessed 21-April-2019].

[12] M. James, "Google says spectre and meltdown are too difficult to fix." I Programmer. https://www.i-programmer.info/news/149-security/12556-google-says-spectre-and-meltdown-are-too-difficult-to-fix.html, Feb 2019. [Online; accessed 21-April-2019].

[13] "Speculative execution side-channel attack ("spectre")." Mozilla. https://www.mozilla.org/en-US/security/advisories/mfsa2018-01/, Jan 2018. [Online; accessed 21-April-2019].

[14] D. Surma, "Meltdown/spectre." Google. https://developers.google.com/web/updates/2018/02/meltdown-spectre, Oct 2018. [Online; accessed 21-April-2019].

[15] "Kernel page-table isolation." Wikipedia. https://en.wikipedia.org/wiki/Kernel_page-table_isolation, May 2018. [Online; accessed 21-April-2019].

[16] "Intel, arm and amd chip scare: What you need to know." BBC. https://www.bbc.com/news/technology-42562303, Jan 2018. [Online; accessed 22-April-2019].

[17] C. Metz and N. Perlroth, "Researchers discover two major flaws in the world's computers." The New York Times. https://www.nytimes.com/2018/01/03/business/computer-flaws.html, Jan 2018. [Online; accessed 22-April-2019].

[18] "Spectre (security vulnerability)." Wikipedia. https://en.wikipedia.org/wiki/Spectre_(security_vulnerability), Apr 2019. [Online; accessed 22-April-2019].