Le rapport d'avancement SDL

Progrès dans la réduction des vulnérabilités logicielles et le développement d'atténuations des menaces chez Microsoft

2004 - 2010

Microsoft

Le rapport d'avancement SDL

Auteurs

David Ladd – *Microsoft Security Engineering Center*
Frank Simorjay – *Microsoft Trustworthy Computing*
Georgeo Pulikkathara – *Microsoft Trustworthy Computing*
Jeff Jones – *Microsoft Trustworthy Computing*
Matt Miller – *Microsoft Security Engineering Center*
Steve Lipner – *Microsoft Security Engineering Center*
Tim Rains – *Microsoft Trustworthy Computing*

Préface

Cette année est marquée par un événement relativement important dans ma carrière : c'est le 40e anniversaire du tout premier rapport que j'ai rédigé concernant la sécurité des logiciels. Alors que je réfléchissais à cette étape décisive, trois choses me sont venues à l'esprit. Premièrement, j'ai réalisé que cela faisait bien longtemps que j'étais dans cette industrie ! Deuxièmement, je constate que de nombreuses approches de conception de logiciels sécurisés n'ont tout simplement pas fonctionné. Troisièmement, je crois que l'industrie a désormais développé des approches efficaces à la conception de logiciels plus sécurisés, et je suis prudemment optimiste pour l'avenir.

J'ai passé la majorité des onze dernières années à travailler dans le groupe Trustworthy Computing de Microsoft, à intégrer des principes de sécurité et de confidentialité à la culture ainsi qu'aux processus de développement de logiciels de la société. Le cycle de développement sécurisé SDL (Security Development Lifecycle) de Microsoft est un élément clé de la directive Informatique de confiance (Trustworthy Computing). Le SDL est un processus d'assurance sécurité centré sur le développement de logiciels et introduisant à la fois la sécurité et la confidentialité à travers toutes les phases du processus de développement. Le SDL est une politique obligatoire dans l'ensemble de l'entreprise depuis 2004. Il combine des approches holistiques et pratiques de la réduction du nombre et de la gravité des vulnérabilités dans les produits et services Microsoft, limitant ainsi la possibilité que des attaques puissent compromettre des ordinateurs. Nous partageons gratuitement le SDL avec l'industrie logicielle et les équipes de développement, et nous sommes heureux de constater qu'il a été adopté (quelquefois sous une forme adaptée) par une variété de vendeurs indépendants de matériel et de logiciels, d'organismes publics et d'équipes de développement des utilisateurs finaux.

Avant même que le SDL ait été formalisé, nous avons créé une petite équipe dédiée à la recherche des vulnérabilités de sécurité, de leurs causes ainsi que de méthodes systématiques permettant de les supprimer ou d'en atténuer les effets. Nous avons appris à nous référer à cette équipe ainsi qu'à ses activités sous le nom de « science de la sécurité ». La science de la sécurité est la « science à l'intérieur du SDL ». Nous étudions la manière dont les systèmes informatiques peuvent être attaqués ainsi que la manière dont ces attaques peuvent être contrecarrées. Nous développons ensuite des outils et techniques de pointe permettant de rendre encore plus difficiles les attaques réussies sur les logiciels. Une fois convaincus que ces outils et techniques sont fiables et efficaces, nous sollicitons leur application en vertu du SDL. Nous les publions ensuite auprès du public de sorte que nos clients, nos partenaires et même nos concurrents puissent concevoir des logiciels plus sécurisés.

Dans ce rapport, vous en apprendrez plus sur l'évolution du SDL ainsi que sur les progrès que nous avons effectués dans l'utilisation du SDL et de la science de la sécurité pour réduire les vulnérabilités et atténuer les menaces envers les logiciels et services Microsoft. Nous pensons que le SDL nous a permis de protéger les clients Microsoft et que, grâce à son large taux d'adoption, il a aussi permis de protéger un nombre conséquent d'internautes.

Si vous êtes un distributeur de logiciels indépendant ou tout autre développeur de logiciels et si vous utilisez le SDL, ce rapport vous fournira des informations de base sur le SDL et la manière dont il a évolué depuis les six dernières années. Si vous n'utilisez pas encore le SDL, nous espérons que ce rapport vous aidera à comprendre pourquoi nous pensons qu'il s'agit d'un processus performant et efficace et vous encouragera à essayer le SDL dans votre propre société.

Steve Lipner
Directeur principal de la stratégie d'ingénierie de sécurité (Security Engineering)
Trustworthy Computing Security, Microsoft

Introduction

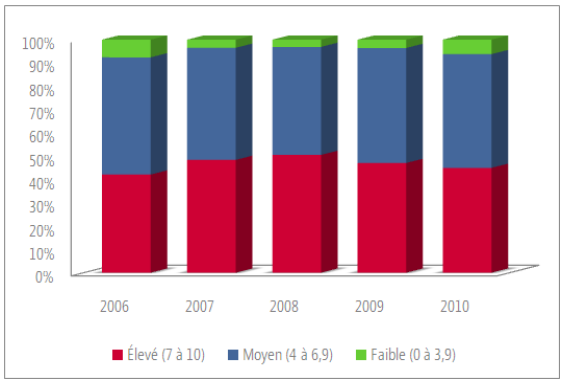
Les vulnérabilités sont des points faibles dans un logiciel qui permettent à une personne malveillante de compromettre l'intégrité, la disponibilité ou la confidentialité dudit logiciel ou des données qu'il traite. Certaines des vulnérabilités les plus graves permettent aux personnes malveillantes d'exécuter du code de leur choix, compromettant potentiellement l'ordinateur, ses logiciels ainsi que les données qui y résident. La divulgation d'une vulnérabilité peut provenir de sources variées, y compris de distributeurs de logiciels, de distributeurs de logiciels de sécurité, de chercheurs en sécurité indépendants, ainsi que de créateurs de logiciels malveillants (en anglais, « malware »).

Il est impossible d'empêcher complètement l'introduction de vulnérabilités lors du développement de projets logiciels à grande échelle. Tant que des êtres humains écriront du code, des erreurs menant à des imperfections du logiciel seront commises : aucun logiciel n'est parfait. Certaines imperfections (les « bogues ») empêchent simplement le logiciel de fonctionner comme prévu, mais d'autres bogues peuvent représenter des vulnérabilités. Les vulnérabilités ne sont pas toutes égales : certaines ne pourront pas être exploitées, car des mesures d'atténuation spécifiques empêchent les personnes malveillantes de les utiliser. Un certain pourcentage des vulnérabilités présentes dans un logiciel donné seront cependant exploitables.

Les données de la base de données National Vulnerability Database[1] démontrent que la tendance à long terme sur la divulgation de vulnérabilités est caractérisée par des milliers de divulgations de vulnérabilités chaque année, la plupart combinant une sévérité élevée et une complexité peu élevée. En outre, la plupart des divulgations de vulnérabilités concernent des applications plutôt que des systèmes d'exploitation ou des navigateurs Web. Cette tendance est inquiétante, car elle signifie qu'il existe des milliers de divulgations de vulnérabilités de haute sévérité dans les applications, et que la plupart d'entre elles sont relativement faciles à exploiter.

[1] Les informations contenues dans cette section ont été compilées à partir des données de divulgation de vulnérabilité publiées dans la National Vulnerability Database (http://nvd.nist.gov), la base de données du gouvernement des États-Unis qui regroupe toutes les données de gestion de vulnérabilités basées sur des normes et identifiées à l'aide du protocole de sécurité Security Content Automation Protocol (SCAP).

Figure 1 : *à gauche* : Divulgations de vulnérabilités dans l'industrie par ordre de sévérité, 2006 à 2010 ; *à droite* : Vulnérabilités dans l'industrie par ordre de complexité d'accès, 2006 à 2010



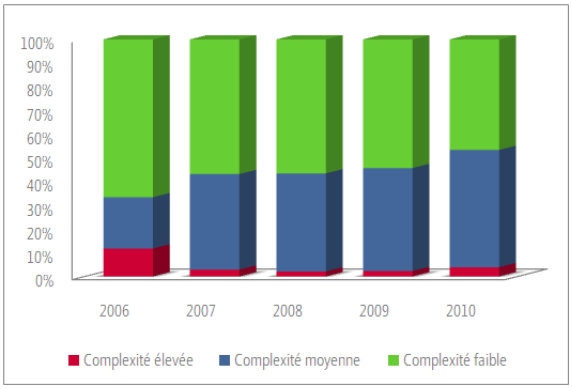
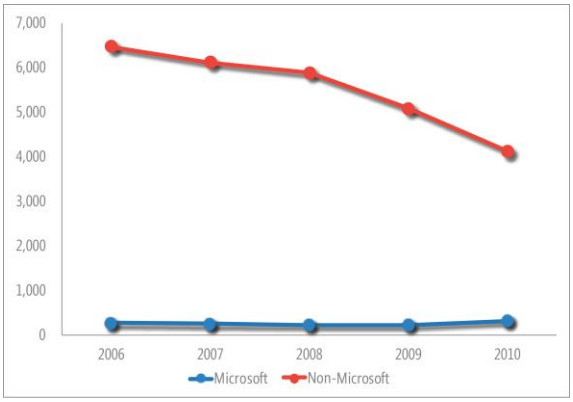
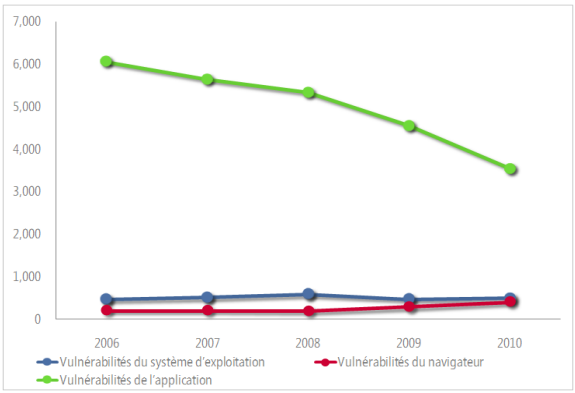


Figure 2 : *à gauche* : Divulgations de vulnérabilités pour les produits Microsoft et non Microsoft, 2006 à 2010 ; *à droite* : Vulnérabilités dans les systèmes d'exploitation, navigateurs et applications à l'échelle de l'industrie, 2006 à 2010

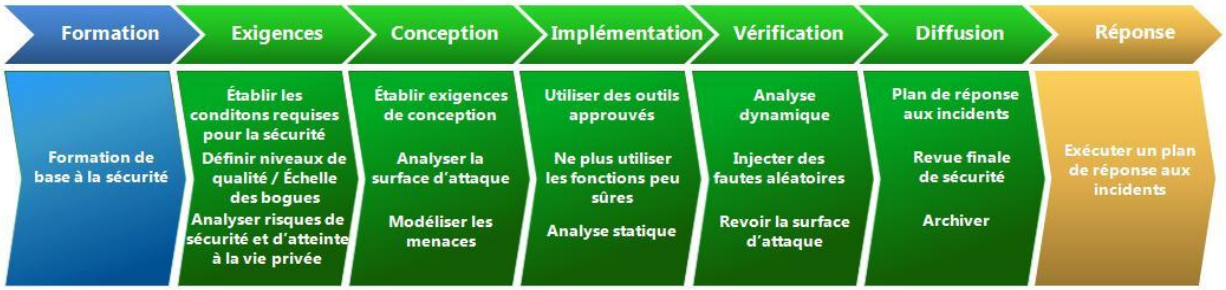




Microsoft pense que les données sensibles et les informations personnelles doivent être protégées, que les sociétés informatiques doivent adhérer à des pratiques commerciales favorisant la confiance, et que l'industrie de la technologie doit se concentrer sur une ingénierie et des pratiques solides pour fournir des produits et des services fiables et sécurisés. Notre approche pour faire face à ces défis est appelée Trustworthy Computing (Informatique de confiance) : un effort collaboratif à long terme permettant de créer et d'assurer à tous une expérience informatique sécurisée, privée et fiable.

Le cycle de développement sécurisé SDL de Microsoft est un élément clé de la directive Informatique de confiance. Le SDL est un processus d'assurance sécurité centré sur le développement de logiciels et introduisant à la fois la sécurité et la confidentialité à travers toutes les phases du processus de développement. Processus appliqué à tous les développements Microsoft depuis 2004, le SDL a joué un rôle important dans la prise en compte de la sécurité et du respect de la vie privée dans les logiciels Microsoft et dans la culture de l'entreprise. Le SDL combine des approches à la fois holistiques et pratiques de la réduction du nombre et de la gravité des vulnérabilités dans les produits et services Microsoft, limitant ainsi la possibilité que des attaques puissent compromettre des ordinateurs. Microsoft partage gratuitement le SDL avec l'industrie logicielle et les équipes de développement, au sein desquelles il a permis de développer des logiciels plus sécurisés.

Figure 3 : Les pratiques qui définissent le processus SDL (Security Development Lifecycle)



La science de la sécurité est la science au cœur du SDL. Elle s'appuie sur une série de recherches innovantes permettant de comprendre la manière dont les systèmes informatiques sont attaqués ainsi que la manière d'empêcher ces attaques ou d'en atténuer les effets. La science de la sécurité produit et développe ensuite des outils et techniques de pointe permettant de rendre encore plus difficiles les attaques réussies sur les logiciels. La science de la sécurité améliore le SDL de trois manières :

- En aidant à repérer les vulnérabilités logicielles.
- En développant des techniques et des outils d'atténuation d'exploitation que les développeurs doivent adopter.
- En surveillant constamment les tendances et l'activité concernant les menaces et en améliorant les outils et processus en fonction de ces observations. Si les efforts de surveillance déterminent qu'une nouvelle menace a fait son apparition dans l'écosystème, les processus de réponse de sécurité de Microsoft sont engagés.

Avantages du développement sécurisé

Les sociétés qui commencent à mettre en place des programmes de développement de logiciels se concentrent d'abord généralement sur la fonctionnalité et ne placent les tests de sécurité qu'à la fin du processus de développement. Cette approche possède cependant d'importants inconvénients par rapport à une approche structurée qui intègre les efforts de sécurité à travers la totalité du processus de développement.

Le National Institute of Standards and Technology (NIST) estime[2] que les corrections de code appliquées après la mise sur le marché peuvent coûter 30 fois plus cher que les corrections effectuées lors de la phase de conception. Des études récentes de Forrester Research, Inc. et de l'Aberdeen Group ont démontré que lorsque les sociétés adoptent un processus structuré tel que le SDL de Microsoft, qui aborde systématiquement la sécurité du logiciel au cours de la phase appropriée du cycle de vie, les vulnérabilités ont plus de chances d'être trouvées et corrigées plus tôt durant le cycle de développement, ce qui permet ainsi de réduire le coût total du développement du logiciel.

En janvier 2011, Forrester a publié[3] les résultats d'une enquête soutenue par Microsoft concernant les principaux influenceurs en matière de développement de logiciels en Amérique du Nord. Forrester a découvert que, bien que la sécurité des applications ne soit pas une pratique courante dans la majorité des cas, ceux qui utilisaient une approche prescriptive coordonnée obtenaient un meilleur retour sur investissement.

---

[2] Rapport NIST 2002 ; *The Economic Impacts of Inadequate Infrastructure for Software Testing*
[3] Forrester Consulting 2011 ; *State of Application Security: Immature Practices Fuel Inefficiencies, But Positive ROI is Attainable*

En août 2010, l'Aberdeen Group a publié[4] les résultats de recherches confirmant que le coût total annuel des initiatives de sécurité des applications était largement moins important que les avantages accumulés. En décembre 2010, l'Aberdeen Group a publié des résultats de recherche plus détaillés concernant des sociétés mettant en œuvre des programmes structurés pour le développement de la sécurité et a « découvert que celles-ci obtenaient un retour important allant jusqu'à 4 fois leur investissement annuel en matière de sécurité des applications ».

En gardant ces informations de base à l'esprit, le reste de ce rapport se concentrera sur les progrès accomplis par Microsoft grâce à l'utilisation du SDL et de la science de la sécurité pour réduire les vulnérabilités et développer des atténuations de menaces dans les logiciels et services Microsoft de façon à protéger les clients Microsoft et les internautes. Les résultats de recherches sur certaines des applications les plus répandues sont inclus pour donner un aperçu du nombre de ces applications qui tirent avantage des atténuations de sécurité incluses dans les systèmes d'exploitation Windows®.

[4] Aberdeen Group 2010 ; *Securing Your Applications: Three Ways to Play*

Security Development Lifecycle

Les débuts de la Sécurité Microsoft (1990 à 2004)

Microsoft possède une longue histoire dans la mise en œuvre de la sécurité logicielle : certains efforts, tels que la conformité des critères communs, s'étendent sur des décennies. Cependant, avant l'adoption du SDL chez Microsoft, les processus de sécurité étaient inégaux : les équipes produit implémentaient généralement les protections de sécurité et de confidentialité à leur propre discrétion. Certaines équipes de développement soumettaient leurs applications à des contrôles considérables, alors que d'autres mettaient en avant les nouvelles fonctionnalités au détriment de la sécurité et de la confidentialité.

Menaces émergentes : la réponse de Microsoft

Une série d'incidents particulièrement visibles concernant des logiciels malveillants dans la seconde moitié des années 1990 (Melissa) et le début des années 2000 (Code Red, Nimda, UPnP, etc.) a amené Microsoft à repenser les processus et la stratégie de sécurité des développeurs. Les éléments précoces de ce qui est devenu le ***processus de modélisation des menaces (par exemple, STRIDE)*** ont été introduits pendant cette période, ainsi que le concept d'une ***échelle des bogues*** universelle permettant d'établir la sévérité d'une vulnérabilité. L'***analyse de la cause principale*** formalisée a été mise en œuvre de manière à comprendre les causes et l'impact de différents types de vulnérabilités à travers la gamme de produits Microsoft. D'autres modifications basées sur la technologie, dont l'***analyse statique du code***, ont été introduites lorsque Microsoft a acquis Intrinsa et son outil d'analyse statique PREfix. De nombreux efforts de sécurité ont été appliqués de manière limitée à Windows 2000 et sont finalement devenus les fondations mêmes du SDL, tel qu'il est mis en pratique aujourd'hui.

STRIDE est le terme utilisé chez Microsoft pour classer les menaces découvertes lors des activités de modélisation des menaces.

- ***S****poofing (usurpation)*
- ***T****ampering (altération)*
- ***R****epudiation (répudiation)*
- ***I****nformation disclosure (divulgation d'informations)*
- ***D****enial of Service (déni de service)*
- *Elevation of privilege (élévation de privilèges)*

Même si ces changements ont eu un impact positif sur les logiciels de Microsoft, ils n'étaient ni obligatoires, ni assez complets pour faire face aux défis attendant les produits Microsoft. La perturbation causée par la publication de logiciels malveillants exploitant les vulnérabilités des logiciels Microsoft a affecté de nombreux clients Microsoft et internautes, et a causé une diminution de la confiance. Bill Gates a par conséquent publié son mémo Trustworthy Computing (Informatique de confiance) en janvier 2002. L'initiation du Trustworthy Computing a radicalement changé les priorités de la société en matière de sécurité logicielle. Ce mandat exécutif a placé la sécurité au sommet de la liste des priorités de Microsoft et a fourni l'élan nécessaire pour une campagne soutenue de modification de la culture de l'ingénierie.

.NET et Windows Security Push

Le soutien au plus haut niveau de l'exécutif a fourni en 2002 aux experts de sécurité Microsoft une opportunité « sanctionnée » de tester l'application d'une approche plus holistique du développement sécurisé. Microsoft a temporairement interrompu le développement du Common Language Runtime (CLR) de .NET Framework pour déplacer l'attention des équipes de développement des fonctionnalités vers l'écriture de code sécurisé. Sur une période d'environ dix semaines, des douzaines de bogues de sécurité ont été repérés et corrigés, et l'accent collectif placé sur la sécurité a introduit de nouvelles méthodes de protection des logiciels, un exemple notable étant la ***réduction de la surface d'attaque***.

L'analyse et la réduction de la surface d'attaque sont une activité d'analyse de sécurité effectuée lors de la phase de conception du SDL de Microsoft. Son but est de réduire la quantité de code et de données exposée aux utilisateurs non fiables.

Le travail sur le .NET CLR s'est révélé être un succès. Par conséquent, le comité de direction a décidé qu'une approche similaire serait utilisée pour la sortie prochaine de Windows Server 2003. Cet effort a été désigné sous le nom de « Windows Security Push ». L'application de ces processus aux efforts de développement de Windows offrait d'impressionnants défis logistiques et technologiques. Par exemple, la taille de la base de code de Windows était (à l'époque) environ dix fois supérieure à celle du CLR. La taille et la complexité organisationnelles de la Division Windows ont aussi poussé vers une dépendance accrue envers la technologie. Pourtant, les innovations de processus ont continué d'émerger. ***La modélisation des menaces basées sur les outils*** en était à ses balbutiements, et en plus de la solution PREfix existante, ***l'analyse statique légère*** (PREfast) a été introduite sur les ordinateurs des développeurs comme moyen de recherche et de correction des bogues avant l'archivage du code. En outre, les ***« audits de sécurité »*** ont été rendus obligatoires. Ils étaient le précurseur de l'***analyse finale de la sécurité (FSR)***, qui est depuis devenue un élément critique du SDL chez Microsoft.

Les Security Push .NET CLR, Windows Server 2003, SQL Server 2000 SP3, et autres ont produit des résultats impressionnants. Les responsables de la sécurité Microsoft ont cependant compris qu'un « push » précédant de peu la sortie d'un produit ne pouvait pas être aussi efficace que l'intégration de la sécurité à l'intérieur même de la conception et du développement du produit. Grâce à l'assurance née de l'expérience « security push », une proposition concernant un processus obligatoire a été présentée en 2004 à l'équipe de direction principale de Microsoft. La proposition a été acceptée en tant que politique Microsoft, le résultat étant que tous les produits et services en ligne exposés à un risque de sécurité significatif ou traitant des données sensibles doivent se conformer aux exigences du SDL.

Microsoft SDL (2004 à aujourd'hui)

Grâce au SDL, Microsoft a choisi une approche incrémentielle de l'amélioration des processus de sécurité, qui intègre les améliorations technologiques et les améliorations des processus de sécurité et de confidentialité au fur et à mesure que celles-ci s'affinent. Comme il est attendu d'un processus qui s'affine au fil du temps, de nombreuses techniques et de nombreux processus de sécurité ont été intégrés assez tôt dans l'évolution du SDL. Au fil du temps, l'accent cumulatif s'est déplacé vers l'efficacité et la productivité des outils et processus SDL, et les améliorations ont été intégrées soit sous forme d'exigences, soit sous forme de recommandations. Certaines additions au SDL abordent de nouvelles menaces alors que d'autres sont des améliorations d'exigences existantes.

De nombreuses techniques et outils de sécurité étaient en cours d'utilisation par les groupes de produit de Microsoft, avant même leur inclusion en tant qu'exigences ou recommandations de sécurité Microsoft SDL

Les éléments suivants, marqués en **gras**, représentent une liste annuelle ***non*** exhaustive des additions au SDL. Il s'agit plutôt d'une liste des additions majeures au SDL. Vous pourrez trouver toute la discussion autour des exigences de Microsoft SDL (depuis la version 3.2) sur le site MSDN Developer Center. De plus, ces éléments indiquent les moments précis où des processus ou technologies sont devenus des ***exigences*** SDL. Il est cependant important de noter que de nombreuses techniques étaient en cours d'utilisation par les groups produits de Microsoft ou étaient incluses dans le SDL en tant que recommandations longtemps avant de devenir des exigences SDL. Dans un souci de concision et de clarté, la chronologie des recommandations SDL a été omise.

2004 – SDL 2.0

La première version officielle du Microsoft SDL, créée en 2004, était essentiellement une liste codifiée d'affinements des techniques et processus précédemment utilisés lors des Security Push .NET et Windows mentionnés précédemment. Certains éléments sont listés en tant qu'exigences, d'autres en tant que recommandations. La version initiale du SDL garantissait que des actions telles que la modélisation de menaces, l'analyse statique et l'Analyse finale de la sécurité étaient obligatoires sur les logiciels Microsoft exposés à des risques de sécurité significatifs.

Le concept de « **risque de sécurité significatif** » sert à déterminer quelles sont les applications assujetties aux contraintes du Microsoft SDL. En résumé, toute application qui présente une ou plusieurs des caractéristiques suivantes doivent se conformer au SDL :

- Applications déployées dans un environnement d'entreprise.
- Applications traitant des informations personnelles ou d'autres informations sensibles.
- Applications surveillant ou interagissant avec un réseau

2005 – SDL 2.1 et 2.2

Échelle de bogues. L'échelle des bogues consiste en un niveau et un seuil de qualité qui s'appliquent à l'ensemble du projet de développement. Elle a été ajoutée au SDL en tant que moyen de configurer des critères de qualité en fonction de la gravité des bogues. Elle est définie au début du projet, ce qui permet de mieux comprendre les risques associés aux problèmes de sécurité. Les équipes savent alors sur quels risques se concentrer pendant le développement. L'équipe projet négocie une échelle de bogues avec son conseiller en sécurité avec des clarifications spécifiques au projet et, si nécessaire, des exigences de sécurité plus strictes définies par le conseiller en sécurité. L'échelle des bogues, une fois définie, n'est jamais assouplie.

Fuzzing (fichier et RPC). L'introduction délibérée de données aléatoires ou invalides dans un programme (processus connu sous le nom de « test de données aléatoires » ou « fuzzing ») a été définie pour aider à trouver les erreurs d'assertion, les fuites de mémoire et les plantages. Cette technique, utilisée avec succès par l'écosystème des utilisateurs malveillants, a été adoptée par Microsoft en tant que pratique de test obligatoire, en raison de son efficacité. Les techniques de recherche d'erreurs se concentraient initialement sur les analyseurs de fichier et sur les interfaces RPC.

Normes de cryptographie. Une série d'exigences concernant l'utilisation de la cryptographie dans les applications Microsoft a été définie pour exiger des équipes de produits qu'elles utilisent des normes établies de cryptographie plutôt que d'essayer de développer leurs propres algorithmes. D'autres exigences stipulent l'utilisation de bibliothèques de cryptographie (telles que CryptoAPI), découragent l'utilisation de solutions de cryptographie intégrées au code (agilité de chiffrement), et spécifient l'utilisation de puissants algorithmes de cryptographie, avec une notification délibérée auprès des utilisateurs s'il a fallu faire appel à un algorithme moins puissant dans un souci de compatibilité.

Tests de vérification au moment de l'exécution. En plus de la recherche d'erreurs, le SDL nécessitait des tests supplémentaires sur les programmes au moment de l'exécution. Grâce à AppVerifier ainsi qu'à d'autres outils, les équipes ont détecté des erreurs dans une application cible en surveillant de manière passive et en établissant un rapport sur le comportement du programme lorsque celui-ci est exécuté dans son environnement opérationnel prévu.

2006 – SDL 3.0 et 3.1

Fuzzing (ActiveX). Les tests de fuzzing ont été étendus aux contrôles ActiveX® inclus dans les applications Microsoft. Les contrôles ActiveX peuvent servir de conduits pour l'injection de données non fiables à partir de sites Web et d'autres emplacements vers un ordinateur hôte. Un dépassement de mémoire tampon dans un contrôle ActiveX scripté peut mener à l'exécution de code en provenance de la zone Internet dans le contexte de l'utilisateur connecté.

API interdites (Banned.h). La bibliothèque runtime C a été initialement créée il y a plus de 25 ans, lorsque la connectivité réseau et l'environnement des menaces représentaient un problème moins important. Microsoft a décidé de retirer un sous-ensemble de la bibliothèque runtime C pour supprimer des fonctions connues pour représenter des risques de sécurité, l'accent étant mis sur les dépassements de mémoire tampon. Cette exigence était initialement formulée sous la forme d'une liste d'API dangereuses, mais elle a évolué au fil du temps pour devenir un fichier d'en-tête (Banned.h) pouvant être utilisé en combinaison avec un compilateur afin de fournir une méthode de nettoyage automatique du code source.

Normes de respect de la vie privée pour le développement. Microsoft a établi d'importantes directives internes à destination des développeurs, centrées sur la protection du client et de la vie privée de l'utilisateur. Ces directives ont permis aux développeurs de comprendre les attentes des utilisateurs, les lois générales régissant la vie privée, ainsi que les procédures autorisées permettant de garantir le respect de la vie privée à travers les produits et les services Microsoft.

Exigences des services en ligne. Avant SDL 3.1, il existait deux ensembles distincts d'exigences de sécurité : le SDL à destination du développement d'applications serveur/client (appliqué aux équipes produit traditionnelles) et un ensemble séparé d'exigences de sécurité appliquées au développement de MSN et d'autres services en ligne. L'exigence de sécurité Online Services a été ajoutée pour unifier les éléments des deux processus en un seul SDL cohérent.

2007 – SDL 3.2

Défenses anti-script de site à site. Le SDL a mandaté des procédures permettant d'empêcher ou d'atténuer les attaques de script de site à site (XSS), y compris la validation des entrées, l'encodage des sorties et la vérification de vulnérabilités en boîte noire. Cette exigence a aussi recommandé l'utilisation de la bibliothèque Anti-XSS pour l'encodage des sorties, qui utilise le principe d'inclusion (aussi connue sous le nom de Listes blanches).

Défenses contre les injections SQL. Microsoft a inclus des conseils SDL spécifiques sur les attaques par injection de code SQL, y compris l'utilisation de procédures stockées et de requêtes paramétrées ainsi que de vérifications de sécurité en boîte noire.

Défenses en matière d'analyse XML. Une exigence a été ajoutée pour faire face aux attaques d'analyse XML. Un service Web essayant généralement d'analyser tout code XML valide qui lui est soumis, il est extrêmement important de rechercher des erreurs dans les interfaces pour garantir une validation correcte des entrées. En plus de la recherche d'erreurs dans l'analyseur, des exigences concernant des versions spécifiques sécurisées des analyseurs syntaxiques XML ont été incluses.

Premier lancement public du processus Microsoft SDL. En réponse aux requêtes des clients, utilisateurs et autres parties intéressées, Microsoft a publié le processus SDL de manière à augmenter la transparence et à permettre une meilleure compréhension des processus et des technologies utilisées lors de la sécurisation des logiciels Microsoft.

2008 – SDL 4.0 et 4.1

Randomisation du format d'espace d'adresse (ASLR). La randomisation du format d'espace d'adresse est passée d'une recommandation à une exigence. Cette exigence a requis l'activation de l'ASLR sur tous les fichiers binaires en code natif (C/C++) pour fournir une protection contre les attaques *return-to-libc*. ASLR est décrit en profondeur dans la section « Atténuations par la science de la sécurité » de ce document.

CAT.NET pour code managé et services en ligne. L'utilisation de CAT.NET a été définie pour garantir que l'analyse de code statique pour le code managé (.NET/C#) est effectuée avant l'archivage du code.

Défenses Cross-site request forgery (CSRF). Une exigence (ViewStateUserKey) a été ajoutée pour garantir l'utilisation de jetons uniques, aléatoires et spécifiques à la session de manière à prévenir les attaques CSRF sur les applications Web implémentées en langages .NET.

2009 – SDL 5.0

Fuzzing (réseau). Les exigences de fuzzing ont été étendues sur toutes les interfaces et tous les analyseurs réseau. Les seuils d'acceptabilité sur les tests de fuzzing ont été augmentés (100 000 itérations réussies).

Revues de sécurité opérationnelle. Toutes les applications prévues pour fonctionner sur les centres de données Microsoft doivent passer une revue de sécurité opérationnelle supplémentaire avant leur déploiement et doivent satisfaire à tous les critères de sécurité de développement du SDL. Bien que l'exigence concernant les revues de sécurité opérationnelle ait été précédemment en place, elle a été intégrée dans le SDL pour garantir que les équipes de service en ligne disposent d'un cadre unifié d'exigences de sécurité.

Exigences de sécurité concernant les licences tierces. Extension des pratiques de développement SDL existantes et des critères de services de sécurité à *tout* le code tiers utilisé sous licence dans les produits et les services Microsoft.

Lancements d'outils externes. Premier lancement public des outils de l'équipe Microsoft SDL.

- L'outil de modélisation des menaces SDL permet aux experts en matières non liées à la sécurité de créer et d'analyser des modèles de menaces.
- SDL Template for Visual Studio® Team System (Classic), un modèle qui intègre automatiquement la politique, les processus et les outils associés à Microsoft SDL Process Guidance version 4.1 directement dans l'environnement de développement logiciel Visual Studio Team System.
- SDL Binscope Binary Analyzer, un outil de vérification qui analyse les fichiers binaires pour garantir qu'ils ont été conçus en conformité avec les exigences et recommandations SDL.
- SDL Minifuzz File Fuzzer, un outil basique de tests de fuzzing, conçu pour permettre de détecter les problèmes pouvant exposer des vulnérabilités de sécurité dans le code de manipulation des fichiers.

2010 – SDL 5.1

Conformité « Sample Code » avec SDL. Les exemples de code (sample code) servent de modèle pour de nombreux projets de développement. Des bogues de sécurité dans un exemple de code signifient généralement des bogues de sécurité dans le code tiers tirant parti de cet exemple. Tous les exemples de code fournis avec les produits Microsoft doivent se conformer à la même échelle de sécurité que nos produits.

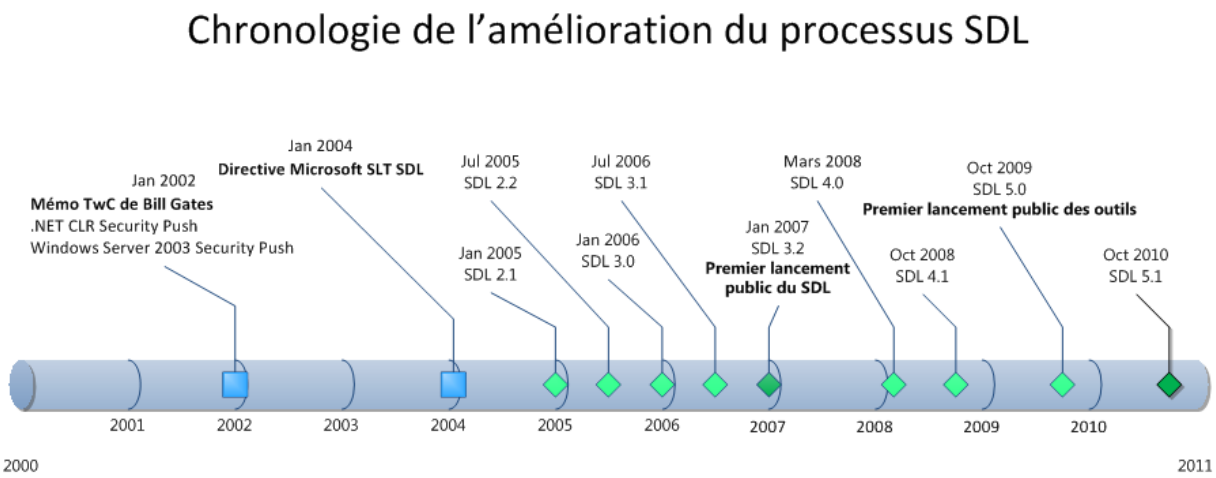
Lancement public de *l'Implémentation simplifiée de Microsoft SDL*. Microsoft SDL est basé sur des concepts de sécurité éprouvés adaptés aux besoins techniques et commerciaux spécifiques à Microsoft. *L'Implémentation simplifiée de Microsoft* SDL est une adaptation simplifiée non propriétaire du Microsoft SDL, conçu pour une utilisation dans d'autres environnements et scénarios de développement ainsi que sur d'autres plateformes logicielles.

Lancements d'outils externes. Deuxième lancement public des outils de l'équipe Microsoft SDL.

- SDL MSF+Agile Template for Visual Studio Team System, un modèle Team Foundation Server incorporant automatiquement la politique, les processus et les outils associés aux conseils de développement SDL for Agile dans le modèle de processus familier Microsoft Solutions Framework (MSF) for Agile software development (MSF-Agile) livré avec Visual Studio Team System.

- Regular Expressions (Regex) Fuzzer, un outil de vérification permettant de tester les expressions régulières à la recherche de vulnérabilités de déni de service.

Figure 4 : Chronologie des étapes décisives dans l'évolution du SDL chez Microsoft



Depuis près d'une décennie maintenant, Microsoft s'appuie sur le SDL et ses précurseurs afin d'introduire des pratiques de sécurité éprouvées dans le développement de ses logiciels. Bien que l'on puisse croire que les méthodologies du SDL et du développement de la sécurité sont « acquises », la réalité n'en est que toute autre. En effet, les menaces logicielles ne sont pas et ne seront jamais statiques. Aussi, nous ne cessons de contrôler l'environnement du danger. Nous effectuons les investissements, en termes de personnel et de technologie, indispensables au perfectionnement et à l'amélioration de Microsoft SDL. Les pratiques recommandées sont aussi largement partagées avec les développeurs de logiciels tiers, afin que chacun d'entre nous puisse bénéficier d'une expérience informatique plus empreinte de sécurité.

Science de la sécurité

Atténuer les risques pour mieux les gérer

Voici un scénario malheureux, mais non moins réaliste : une vulnérabilité de l'un de vos produits a été constatée et divulguée publiquement. Aucun correctif n'est encore disponible. Comment pouvez-vous aider vos clients à s'en prémunir ? En votre qualité de développeur, la réponse à cette question doit être impérativement assimilée au mieux, avant même qu'une telle situation ne se produise. En anticipant ce problème, vous avez la possibilité d'intégrer à votre logiciel des fonctionnalités de défense en profondeur. Ces dernières permettront plus tard de gérer les risques avec plus d'efficacité, lorsque les clients rencontreront une vulnérabilité non corrigée ou inconnue. D'ordinaire appelées *atténuations*, ces fonctionnalités de défense en profondeur sont à même d'affaiblir, partiellement ou complètement, les risques liés à une vulnérabilité.

Diverses approches peuvent souvent être appliquées en vue de pallier une vulnérabilité. Par exemple, une vulnérabilité localisée dans un service réseau peut être atténuée, entre autres, grâce à la protection de la connectivité (pare-feu) et de l'accès (authentification/autorisation), à la désactivation du service ou des fonctionnalités vulnérables (configuration), à la limitation des possibilités d'intrusion (mise en quarantaine). Dans tous les cas, l'objectif demeure le même : empêcher une personne malveillante d'exploiter en toute aisance une vulnérabilité. Les atténuations capables de répondre à cet objectif sont également en mesure de protéger les clients, le temps qu'une mise à jour de sécurité soit développée et mise en œuvre.

Il est important de faire mention d'une approche toute particulière, visant également à atteindre cet objectif. Celle-ci consiste à démembrer les *techniques d'exploitation* sur lesquelles s'appuient les personnes malveillantes lorsqu'ils prévoient de profiter d'une vulnérabilité. Développées par les personnes malveillantes, les techniques d'exploitation peuvent être considérées comme les outils utilisés par ces derniers en vue de transformer une vulnérabilité en un instrument leur permettant d'exécuter les codes de façon arbitraire, et de prendre ainsi le contrôle de l'ordinateur d'un utilisateur. Le démembrement ou la déstabilisation de ces techniques permet en outre de supprimer un élément de la boîte à outils de la personne malveillante. Dans le meilleur des cas, ce processus empêche d'exploitation ou accroît les dépenses, en termes de coûts et de temps, relatives à celle-ci. D'ici à ce qu'un correctif soit développé et mis en œuvre, de tels efforts influent directement sur l'intérêt économique qui motive une personne malveillante à exploiter une vulnérabilité, tout en continuant de protéger les clients. Les atténuations qui constituent cette approche sont généralement désignées sous le nom d'*atténuations d'exploitations*.

Elles présentent d'autres caractéristiques qui rendent cette stratégie d'atténuation intéressante. Il existe plusieurs cas dans lesquels il est possible d'intégrer à une application la logique requise pour démembrer une technique d'exploitation. Grâce à l'utilisation de ce type d'approche, les clients ne s'encombrent aucunement de la mise à niveau d'une atténuation localisée dans leur environnement actuel, lorsqu'une nouvelle vulnérabilité est identifiée. Les atténuations d'exploitations visent à démembrer les techniques d'exploitation. Leur exécution est donc généralement transparente aux fonctionnalités propres des applications, créant ainsi un avantage supplémentaire en faveur de la stratégie. Du fait de leur transparence, il est possible d'activer par défaut les atténuations d'exploitations, sans pour autant entraver les fonctionnalités de l'application. Grâce à l'intégration des atténuations d'exploitations et à l'activation par défaut de celles-ci, les clients ont la possibilité de bénéficier d'une protection générique contre les vulnérabilités connues et celles encore inconnues.

Compte tenu de ces avantages, il n'est pas étonnant qu'au cours des dix dernières années, Microsoft se soit appuyée sur la science de la sécurité pour développer une large gamme de technologies d'atténuation d'exploitations. Ces technologies sont désormais intégrées aux outils de développement tels que Visual Studio. Elles sont également intégrées aux systèmes d'exploitation Windows. Ce niveau d'intégration permet à Microsoft, ainsi qu'aux développeurs de logiciels tiers, de composer des applications comportant des atténuations activées par défaut.

Tactiques

Jusqu'ici, nous avons seulement abordé l'importance des atténuations d'exploitations. Cette section a pour but de vous permettre de mieux comprendre le mode de fonctionnement des atténuations d'exploitations. Pour ce faire, cette partie examine les rouages de certaines technologies spécifiques et tactiques de base, utilisées pour démembrer les techniques d'exploitation. L'ensemble des tactiques utilisées dans les atténuations d'exploitations peut se regrouper en différentes catégories, notamment : l'application d'invariants, l'ajout d'une diversité artificielle et l'exploitation des déficits en matière de connaissances. Les paragraphes qui suivent illustrent le mode de fonctionnement de ces tactiques en mettant l'accent sur des exemples typiques de technologies d'atténuation d'exploitations utilisées. Pour être plus efficaces, les atténuations d'exploitations combinent souvent une ou plusieurs tactiques.

Application d'invariants

L'application de nouveaux invariants est une tactique pouvant être utilisée pour démembrer les techniques d'exploitation. Elle permet de rendre non valides les hypothèses implicites d'une ou de plusieurs techniques d'exploitation. Cette tactique peut être comparée à l'ajout de barreaux à une fenêtre : si auparavant les personnes malveillantes avaient la possibilité d'ouvrir une fenêtre et d'y accéder en toute aisance, elles doivent désormais faire face aux barreaux qui la protègent. Cette idée simple a été intégrée dans la conception de plusieurs technologies d'atténuation, deux des plus importantes étant la *Prévention de l'exécution des données* (DEP) et la protection *Structured Exception Handler Overwrite Protection* (SEHOP).

Prévention de l'exécution des données (DEP)

La plupart du temps, les personnes malveillantes voient les failles comme des données pouvant être exécutées en tant que code. Cette hypothèse découle d'une pratique ordinaire des personnes malveillantes consistant à injecter des codes machines personnalisés (plus communément appelés *shellcode*), pour ensuite les exécuter. Ce processus est connu comme étant une exécution arbitraire de code. Dans la plupart des cas, ce type de code sera enregistré dans la pile ou le segment de mémoire qui constituent les parties de la mémoire d'un programme qui ne contient en principe que des données. Cette technique d'exploitation a souvent été utilisée, étant donné que les anciens processeurs Intel et les versions de Windows antérieures à Windows XP Service Pack 2 ne prenaient pas en charge la mémoire non exécutable. L'intégration de la fonctionnalité de sécurité DEP dans Windows XP Service Pack 2 a mis en place un nouvel invariant qui, à son tour, a permis d'empêcher l'exécution des données en tant que code. Lorsque la fonctionnalité de sécurité DEP est activée, il devient impossible d'injecter et d'exécuter du code machine personnalisé directement depuis les zones de mémoire destinées aux données.

Structured Exception Handler Overwrite Protection (SEHOP)

Certains types de vulnérabilités peuvent permettre à une personne malveillante d'utiliser une technique d'exploitation connue sous le nom de *Structured Exception Handler Overwrite* (réécriture de SEH). Cette technique implique l'altération d'une structure de données utilisée dans la gestion des conditions exceptionnelles susceptibles de se produire au cours de l'exécution d'un programme. L'altération de cette structure de données vise à permettre aux personnes malveillantes d'exécuter le code à partir de toute partie de la mémoire. La fonctionnalité de sécurité SEHOP atténue cette technique, en procédant à des vérifications qui garantissent l'intégrité des structures de données utilisées dans la gestion des exceptions. Grâce à ce nouvel invariant, il est possible de repérer l'altération produite lorsque la technique de réécriture de SEH est utilisée et de démembrer ce type d'exploitation. Si la fonctionnalité de sécurité SEHOP est une technologie d'atténuation encore relativement récente, elle deviendra très certainement une exigence du SDL dans les versions à venir.

Ajouter une diversité artificielle

La diversité au sein d'une population permet de réduire le nombre d'hypothèses universelles pouvant être émises au sujet de ses membres. La biodiversité est souvent utilisée pour traduire ceci : si un nouveau virus survient, la biodiversité permet de garantir que l'ensemble de la population ne soit pas touché. Ce principe est tout aussi pertinent dans le monde numérique, dans la mesure où les personnes malveillantes supposent que la configuration d'un ordinateur reflète celle d'un autre. Il est donc possible d'invalider ces hypothèses et, ce faisant, d'empêcher une personne malveillante d'exploiter efficacement une vulnérabilité, en intégrant une diversité artificielle dans les ordinateurs. Un bon exemple de l'ajout d'une diversité artificielle peut être constaté dans le contexte d'une atténuation d'exploitation appelée randomisation du format d'espace d'adresse (ASLR).

Randomisation du format d'espace d'adresse (ASLR)

Les personnes malveillantes supposent généralement que certains objets, les DLL par exemple, seront placés à la même adresse dans la mémoire, chaque fois qu'un programme est exécuté, et sur chacun des ordinateurs sur lequel ce programme est exécuté. Souvent essentielles au bon déroulement de l'exploitation de la faille, de telles hypothèses sont utiles aux personnes malveillantes. L'incapacité des personnes malveillantes à coder en dur ces adresses permet de rendre complexe, voire impossible, l'écriture d'un code susceptible de faire obstacle à chacun des ordinateurs. La randomisation du format d'espace d'adresse est motivée par cette approche. C'est à l'aide de l'introduction de diversité dans le format d'espace d'adresse d'un programme que la randomisation du format d'espace d'adresse est en mesure de démembrer un grand nombre de techniques d'exploitation. En d'autres termes, la randomisation du format d'espace d'adresse permet de rendre aléatoire l'emplacement des objets dans la mémoire, ceci afin d'empêcher une personne malveillante de présumer efficacement de leur emplacement. Cette tactique a pour effet de définir un format d'espace d'adresse de programme différent sur l'ensemble des ordinateurs. Elle est également utilisée en dernier recours pour empêcher une personne malveillante d'établir des hypothèses universelles relatives à l'emplacement des objets dans la mémoire.

Tirer parti des déficits en matière de connaissances

Dans certaines situations, il est possible de démembrer les techniques d'exploitation en tirant avantage des secrets encore inconnus des personnes malveillantes, ou que ces dernières ne peuvent pas aisément prévoir. Cette tactique peut être comparée à une porte dotée d'une serrure à combinaison. Grâce aux probabilités de combinaison, les personnes malveillantes ne sont pas en mesure d'ouvrir la porte en toute aisance, pour la simple raison qu'il leur est impossible de deviner la combinaison dans un laps de temps raisonnable. Ce concept s'applique tout autant lorsqu'il s'agit de démembrer des techniques d'exploitation. Dans la pratique, l'utilisation de cette tactique est clairement démontrée par la prise en charge du générateur de code sécurisé (Code Generation Security ou GS) que l'on retrouve dans le compilateur Visual C++ de Microsoft.

GS

En termes de vulnérabilité logicielle, le dépassement de mémoire tampon basée sur la pile est un exemple bien connu. L'exploitation de ce type de vulnérabilité se traduit couramment par la réécriture des données critiques utilisées pour exécuter un code, à la fin d'une fonction. Depuis la version Visual Studio 2002, la prise en charge du commutateur de compilateur /GS a été intégrée à Microsoft Visual C++. Lorsqu'il est activé, le commutateur de compilateur pratique une vérification de sécurité supplémentaire dans le but d'atténuer cette technique d'exploitation. Le mode de fonctionnement de cette tactique consiste à placer une valeur aléatoire, plus communément connue sous le nom de cookie, avant la donnée critique elle-même qu'une personne malveillante est susceptible de vouloir remplacer. Au terme de la fonction, le cookie est ensuite vérifié afin de s'assurer que sa valeur est égale à la valeur prévue. En cas de discordance, l'on suppose que l'altération s'est produite et que le programme peut prendre fin en toute sécurité. Par ce concept simple, il est possible de démontrer qu'utilisée d'une façon particulière, une valeur secrète (ici le cookie) est en mesure de démembrer certaines techniques d'exploitation grâce à l'identification d'altération en des points stratégiques du programme. D'une manière générale, il est difficile à la personne malveillante de surmonter le déficit en matière de connaissances induit par le caractère secret de la valeur.

Disponibilité

Les diagrammes de la Figure 5 et de la Figure 6 schématisent la disponibilité de la technologie d'atténuation d'exploitation en termes de versions de système d'exploitation Windows, à commencer par Windows XP RTM[5] et Windows Server® 2003 RTM. Les atténuations d'exploitations « OptIn » indiquent la désactivation par défaut de la fonctionnalité. Dans ce cas, les applications indépendantes doivent activer la fonctionnalité de manière explicite (OptOut indique l'inverse).

[5] RTM est l'abréviation de « released to manufacturing » (version finale) signifiant essentiellement qu'aucun service pack n'a été installé

Figure 5 : Disponibilité des atténuations d'exploitations sur les références client Windows.

	XP RTM, SP1	XP SP2	XP SP3	Vista RTM	Vista SP1	Vista SP2	Win7 RTM
SEH							
SafeSEH	n	o	o	o	o	o	o
SEHOP	n	n	n	n	OptIn	OptIn	OptIn
Prise en charge OptIn par processus SEHOP	n	n	n	n	n	n	o
Segment de mémoire							
suppression de lien sécurisée	n	o	o	o	o	o	o
cookies du titre du bloc	n	o	o	o	o	o	o
suppression de la disponibilité/liste libre	n	n	n	o	o	o	o
chiffrement de métadonnées	n	n	n	o	o	o	o
terminate-on-corruption (application 32 bits)	n	n	n	OptIn	OptIn	OptIn	OptIn
terminate-on-corruption (application 64 bits)	n	n	n	OptOut	OptOut	OptOut	OptOut
DEP							
Prise en charge NX (i386)	n	OptIn	OptIn	OptIn	OptIn	OptIn	OptIn
Prise en charge NX (amd64, application 32 bits)	n	OptIn	OptIn	OptIn	OptIn	OptIn	OptIn
Prise en charge NX (amd64, application 64 bits)	n	AlwaysOn	AlwaysOn	AlwaysOn	AlwaysOn	AlwaysOn	AlwaysOn
Randomisation du format d'espace d'adresse							
prise en charge de la randomisation							
images	n	n	n	OptIn	OptIn	OptIn	OptIn
blocs empilés	n	n	n	OptIn	OptIn	OptIn	OptIn
segments de mémoire	n	n	n	o	o	o	o
PEB/TEB	n	o	o	o	o	o	o
entropie (bits)							
images	0	0	0	8	8	8	8
blocs empilés	0	0	0	14	14	14	14
segments de mémoire	0	0	0	5	5	5	5
PEB/TEB	0	4	4	4	4	4	4
API							
Prise en charge de SetProcessDEPPolicy	n	n	o	n	o	o	o

Figure 6 : Disponibilité des atténuations d'exploitations sur les références serveur Windows.

	Server 03 RTM	Server 03 SP1, SP2	Server 08 RTM	Server 08 R2 RTM
SEH				
SafeSEH	n	o	o	o
SEHOP	n	n	OptOut	OptOut
Prise en charge OptIn par processus SEHOP	n	n	n	o
Segment de mémoire				
suppression de lien sécurisée	n	o	o	o
cookies du titre du bloc	n	o	o	o
suppression de la disponibilité/liste libre	n	n	o	o
chiffrement de métadonnées	n	n	o	o
terminate-on-corruption (application 32 bits)	n	n	OptIn	OptIn
terminate-on-corruption (application 64 bits)	n	n	OptOut	OptOut
DEP				
Prise en charge NX (i386)	n	OptOut	OptOut	OptOut
Prise en charge NX (amd64, application 32 bits)	n	OptOut	OptOut	OptOut
Prise en charge NX (amd64, application 64 bits)	n	AlwaysOn	AlwaysOn	AlwaysOn
Prise en charge NX (ia64)	S/O	AlwaysOn	AlwaysOn	AlwaysOn
Randomisation du format d'espace d'adresse				
prise en charge de la randomisation				
images	n	n	OptIn	OptIn
blocs empilés	n	n	OptIn	OptIn
segments de mémoire	n	n	o	o
PEB/TEB	n	o	o	o
entropie (bits)				
images	0	0	8	8
blocs empilés	0	0	14	14
segments de mémoire	0	0	5	5
PEB/TEB	0	4	4	4
API				
Prise en charge de SetProcessDEPPolicy	n	n	o	o

Adoption d'atténuation

Pour que les technologies d'atténuation décrites précédemment soient efficaces, il est impératif que les applications s'exécutant sur Windows les adoptent. En d'autres termes, les applications doivent intégrer les atténuations de compilateur activées (telles que GS), et également prendre en charge de façon adéquate les atténuations de plateforme, telles que la fonctionnalité de sécurité DEP, ou encore la randomisation du format d'espace d'adresse. Si l'activation d'une ou plusieurs atténuations est un échec, les personnes malveillantes pourront exploiter plus facilement les vulnérabilités, en s'aidant d'outils et de techniques auxquels ils n'auraient pas eu recours en d'autres circonstances.

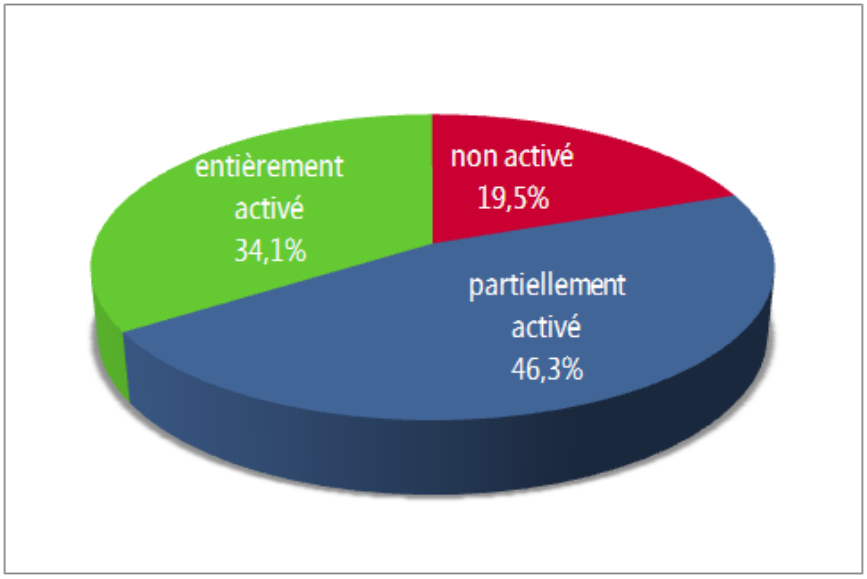
Une question émane de ces implications : si l'efficacité dépend de l'adoption, combien d'applications utilisent à l'heure actuelle ces technologies d'atténuation ? Afin de mieux comprendre la réponse à cette question, nous avons recensé les paramètres de la fonctionnalité de sécurité DEP et de la randomisation du format d'espace d'adresse des versions les plus récentes de 41 applications parmi les plus répandues à travers le monde et utilisées par des millions de personnes. Au terme de cette étude, il a été observé que 71 % des applications recensées prenaient entièrement en charge la fonctionnalité de sécurité DEP. Cependant, seuls 34 % des applications prenaient en charge la randomisation du format d'espace d'adresse. Une analyse en profondeur de ces données est présentée dans les sections qui suivent.

Adoption de la randomisation du format d'espace d'adresse

Avant d'examiner les données relatives à l'adoption de la randomisation du format d'espace d'adresse, il est primordial de comprendre la manière dont les applications prennent effectivement en charge l'ASLR. Afin de permettre la prise en charge de la randomisation du format d'espace d'adresse, une application doit lier ses images exécutables (telles que EXE ou DLL) à l'indicateur /DYNAMICBASE. Cet indicateur permet de signaler aux versions concernées des systèmes d'exploitation Windows la prise en charge de la randomisation du format d'espace d'adresse par une image. Windows ne procède pas à la lecture aléatoire des images qui ne sont pas associées à ce type d'indicateur au moment de l'exécution d'une application. De ce fait, il est probable qu'elles soient transférées à la même adresse. Cette absence de randomisation réduit de manière considérable l'efficacité de la randomisation du format d'espace d'adresse, car elle fournit avantageusement à la personne malveillante une implantation prévisible dans l'espace d'adresse du programme qu'elle est susceptible d'utiliser en vue de développer une attaque. Afin d'optimiser l'efficacité de la randomisation du format d'espace d'adresse, il convient donc de lier à cet indicateur *chacune* des images exécutables se trouvant dans une application. Bien que l'indicateur /DYNAMICBASE soit activé par défaut dans Visual Studio 2010, l'activation de cet indicateur doit être effectuée explicitement dans les paramètres de projet des développeurs pour les versions antérieures de Visual Studio. Ces paramètres par défaut ont été conçus afin de mieux garantir la compatibilité des applications qui établissent des hypothèses sur le format d'espace d'adresse d'un programme.

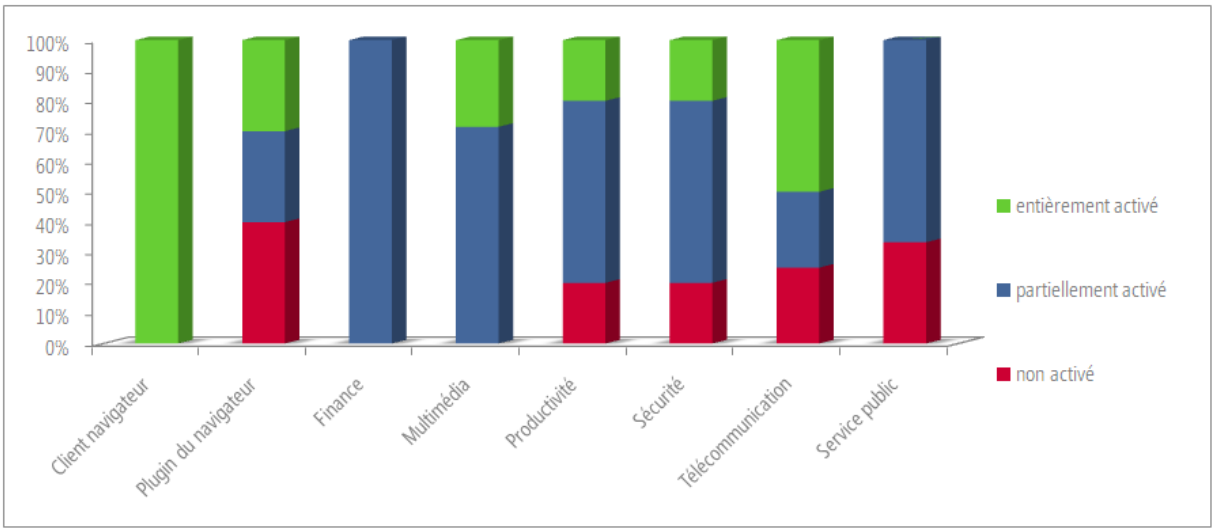
En pratique, sur les 41 applications recensées : 34 % ont procédé à l'activation intégrale de la prise en charge de la randomisation du format d'espace d'adresse, 46 % ont activé partiellement la prise en charge, tandis que 20 % ne l'ont activé pour aucune de leurs images (Figure 7). Ces données nous permettent de constater qu'à ce jour, un grand nombre d'applications consommateurs répandues n'ont pas encore activé intégralement la prise en charge de la randomisation du format d'espace d'adresse.

Figure 7 : Pourcentage des applications pour lesquelles la randomisation du format d'espace d'adresse a été intégralement activée, partiellement activée ou n'a pas été activée.



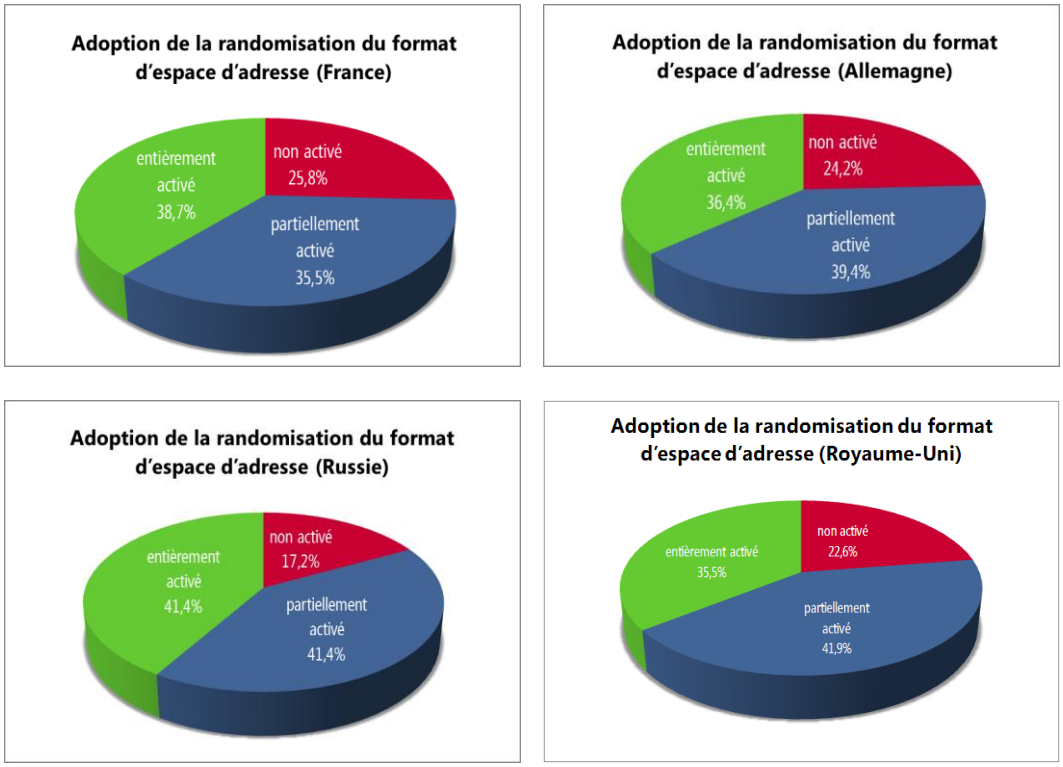
Le diagramme de la Figure 8 présente un aperçu supplémentaire de l'adoption de la randomisation du format d'espace d'adresse, en fonction du segment de marché auquel appartient chacune des applications. D'importantes conclusions peuvent être tirées de ces données. La première étant que l'ensemble des clients navigateurs Web recensés (tels que Windows Internet Explorer®) ont activé intégralement la prise en charge de la randomisation du format d'espace d'adresse. Ce qui n'est malheureusement pas le cas de 70 % des plugins de navigateur recensés, ce qui implique que la présence de plugins de réduit vraisemblablement l'efficacité de la randomisation du format d'espace d'adresse. Seconde conclusion : seul un produit de sécurité sur cinq inclus dans cette analyse a activé intégralement la prise en charge de la randomisation du format d'espace d'adresse. Il est donc important d'en déduire que les produits de sécurité sont intrinsèquement exposés à des données non fiables. Ainsi, l'adoption limitée de la randomisation du format d'espace d'adresse favorise les attaques de vulnérabilités des produits de sécurité perpétrées par les personnes malveillantes.

Figure 8 : Pourcentage par segment de marché des applications pour lesquelles la randomisation du format d'espace d'adresse a été intégralement activée, partiellement activée ou n'a pas été activée.



La question de la pertinence géographique de ces données a aussi été minutieusement étudiée, en prenant en compte un sous-ensemble d'applications recensées et connues pour être particulièrement répandues en France, en Allemagne, en Russie et au Royaume-Uni (avec un total respectif de 31, 33, 29 et 31 applications). Ces résultats sont reportés dans la Figure 9.

Figure 9 : Pourcentage des applications recensées pour lesquelles la randomisation du format d'espace d'adresse a été intégralement activée, partiellement activée ou n'a pas été activée (France, Allemagne, Russie et Royaume-Uni).



Ces données montrent précisément que dans la majorité des segments de marché, l'adoption de la randomisation du format d'espace d'adresse a été marquée par une lenteur, en dépit d'une technologie disponible depuis plus de quatre ans. Elles montrent également que le degré de sécurité de nombre d'applications est plus faible qu'il ne le faudrait, permettant ainsi aux personnes malveillantes d'exploiter plus aisément les vulnérabilités. Il apparaît présentement que les clients navigateurs Web et les logiciels de télécommunication, bien qu'à un niveau moindre, font exception à cette règle. S'il n'y a rien d'étonnant au fait que ces applications aient adopté aussi promptement la randomisation du format d'espace d'adresse, la raison en est que ces types de produits sont directement exposés à une forte quantité de données non fiables retrouvées sur Internet.

Appel à l'action :

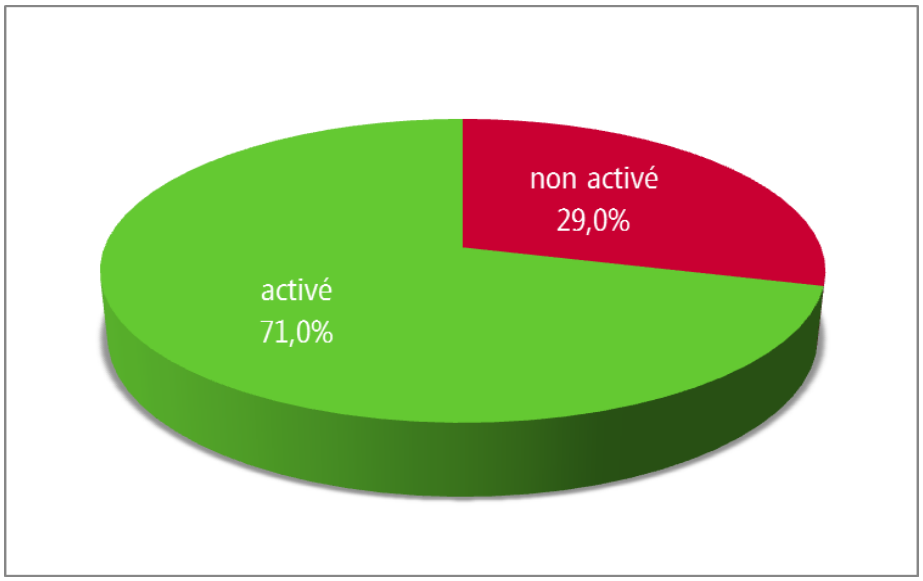
- Les développeurs de logiciels se doivent de garantir que leurs applications sont configurées pour intégrer l'indicateur d'éditeur de liens /DYNAMICBASE.
- Les utilisateurs se doivent à leur tour d'exiger des distributeurs de logiciels qu'ils intègrent à leurs applications l'indicateur d'éditeur de liens /DYNAMICBASE.
- Les paramètres /DYNAMICBASE pour les applications en question peuvent être vérifiés à l'aide de l'outil Microsoft SDL BinScope.

Adoption de la fonctionnalité de sécurité DEP

À la différence de la randomisation du format d'espace d'adresse qui, elle, est activée sur une base de fichier par image, la fonctionnalité de sécurité DEP est activée sur une base par processus. La fonctionnalité de sécurité DEP est activée automatiquement sur toutes les applications 64 bits. Elle doit cependant être activée explicitement sur les applications 32 bits s'exécutant sur les versions client des systèmes d'exploitation Windows. La prise en charge de la DEP doit être activée explicitement sur les applications 32 bits, étant donné que les logiciels n'ont pas été conçus en prenant en compte cette protection. Il existe différents moyens d'activer la protection DEP sur une application, à savoir, entre autres, la liaison des fichiers EXE à l'indicateur /NXCOMPAT et l'appel de l'API SetProcessDEPPolicy au moment de l'exécution. L'activation de la DEP devient permanente, une fois exécutée via l'une de ces procédures. En d'autres termes, il est impossible de désactiver la DEP sur une application, lorsque celle-ci est en cours d'exécution. Du fait que la DEP est activée sur une base par processus, il n'est pas nécessaire de définir d'indicateur spécial destiné à chaque image exécutable.

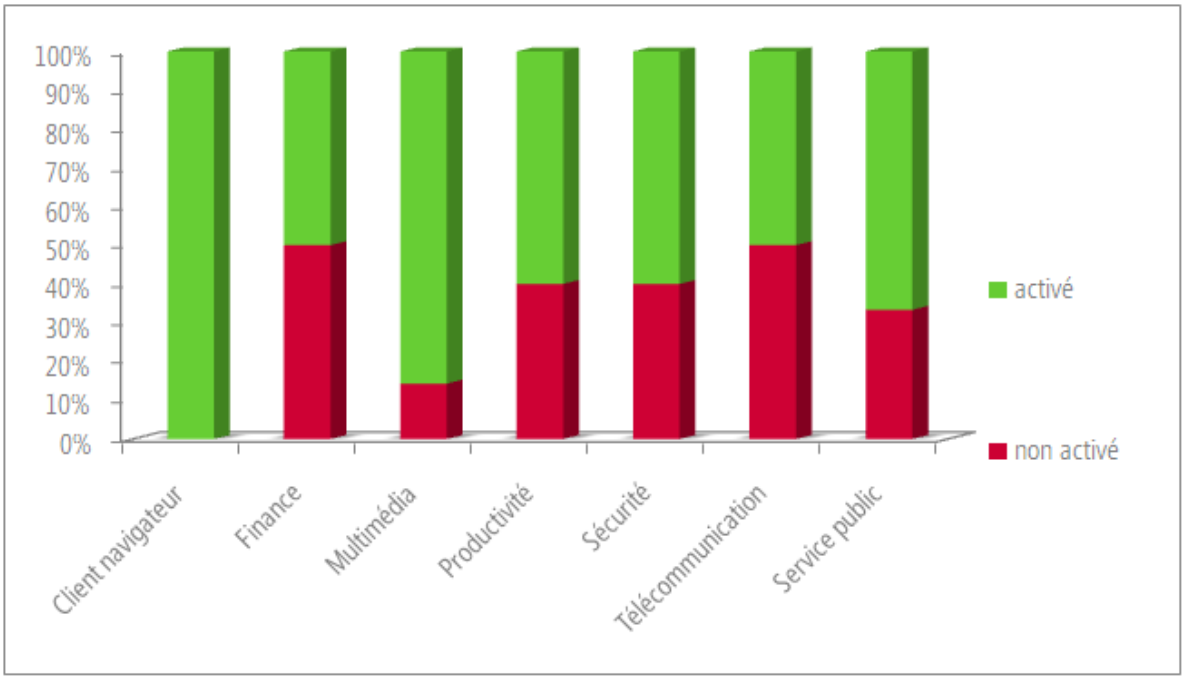
La simplicité de l'activation de la DEP sur une application a assurément contribué à l'accélération de l'adoption de cette fonctionnalité par les distributeurs de logiciels. Le diagramme de la Figure 10 indique que, sur les applications recensées, 71 % ont activé la DEP contre 29 %. Deux facteurs sont à l'origine de ces résultats indiquant que la majorité des applications recensées ont activé la DEP. Le premier se caractérise par la disponibilité de la DEP dès Windows XP Service Pack 2, tandis que la randomisation du format d'espace d'adresse n'est disponible que depuis Windows Vista®. Les développeurs de logiciels ont bénéficié de plus de temps pour adopter la DEP en tant que technologie et, à ce titre, déjouer tous les obstacles rencontrés. Le second facteur s'explique par le processus d'activation relativement simple de la DEP sur une application, dans bien des situations. Il est à noter que les plugins de navigateur ne sont pas inclus dans ces données, pour la raison que leurs paramètres DEP sont indépendants de la configuration du navigateur hôte.

Figure 10 : Pourcentage des applications recensées pour lesquelles la fonctionnalité de sécurité DEP a été activée.



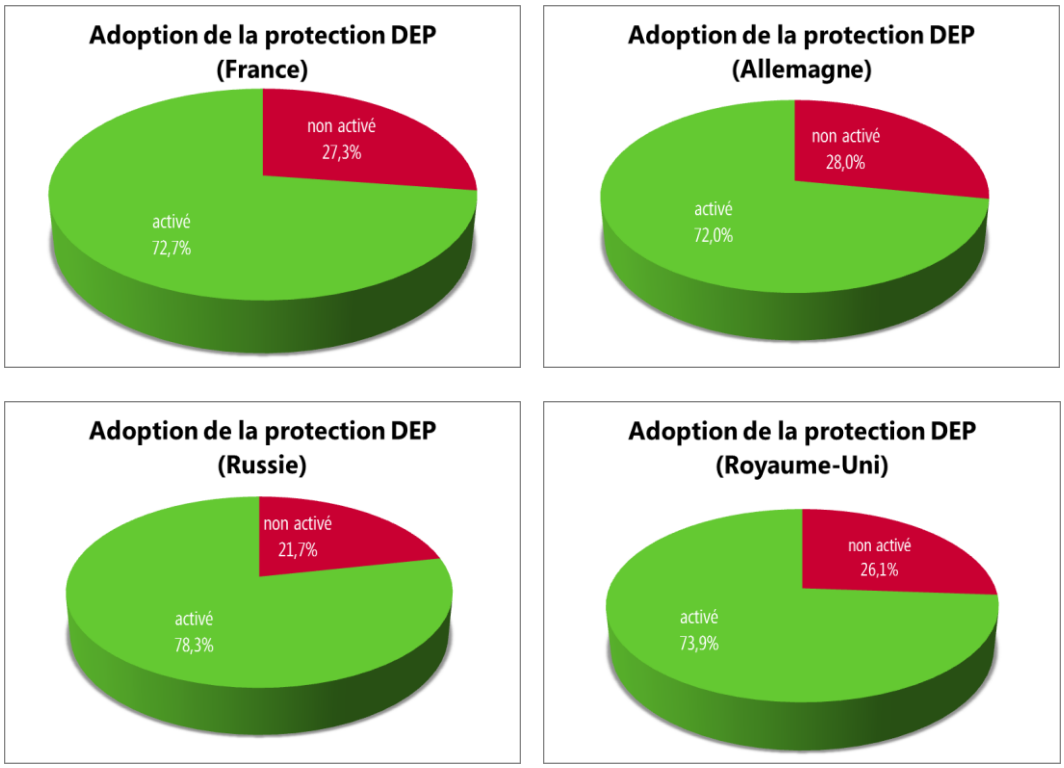
Le diagramme de la Figure 11 présente les détails de ces données, en fonction du segment de marché auquel appartient chacune des applications. À l'image des données relatives à l'adoption de la randomisation du format d'espace d'adresse, les données indiquent que l'ensemble des clients navigateurs recensés a activé intégralement la prise en charge de la fonctionnalité de sécurité DEP. De même, plus de la moitié des applications de chaque segment de marché ont activé la prise en charge de la DEP.

Figure 11 : Pourcentage par segment de marché des applications pour lesquelles la fonctionnalité de sécurité DEP a été ou n'a pas été activée.



La question de la pertinence géographique de ces données a également été minutieusement étudiée, en prenant en compte un sous-ensemble d'applications recensées et connues pour être particulièrement répandues en France, en Allemagne, en Russie et au Royaume-Uni (avec un total respectif de 22, 25, 23 et 23 applications, à l'exception des plugins de navigateur). Ces résultats sont reportés dans la Figure 12.

Figure 12 : Pourcentage des applications recensées en France, en Allemagne, en Russie et au Royaume-Uni, pour lesquelles la fonctionnalité de sécurité DEP a été ou n'a pas été activée.



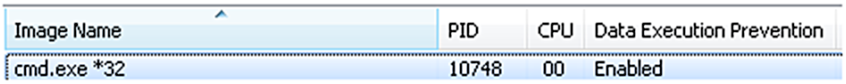
Appel à l'action :

- Les développeurs de logiciels se doivent de garantir que leurs applications sont configurées pour activer la fonctionnalité de sécurité DEP via SetProcessDEPPolicy, ou par l'indicateur d'éditeur de liens NXCOMPAT.

- Les utilisateurs se doivent à leur tour d'exiger des distributeurs de logiciels qu'ils activent la fonctionnalité de sécurité DEP dans leurs applications.

- Les paramètres DEP pour les applications en question peuvent être vérifiés à l'aide de la boîte à outils Enhanced Mitigation Experience Toolkit, de Sysinternals Process Explore (EMET), ou encore du Gestionnaire des tâches de Windows intégré, comme décrit ci-après :

Figure 13 : Gestionnaire des tâches de Windows



Activation des atténuations dans votre logiciel

Ce document explique comment les atténuations peuvent représenter un outil précieux pour limiter les risques encourus face aux vulnérabilités tant connues qu'inconnues. Microsoft mise sur le potentiel des technologies d'atténuation d'exploitations et a ajouté des exigences obligatoires dans le SDL, stipulant qu'il est impératif pour les équipes produit d'user de ces fonctionnalités lors du développement de logiciels. Pour que ces atténuations puissent déployer leur efficacité, il est impératif que les applications s'exécutant sur les systèmes d'exploitation Windows les adoptent. Le recensement des applications consommateurs répandues nous a permis de constater que bien qu'un grand nombre d'applications aient activé la fonctionnalité de sécurité DEP, il n'en est pas de même pour la grande majorité en termes d'activation intégrale de la randomisation du format d'espace d'adresse. Dans l'optique d'améliorer la situation, les distributeurs de logiciels doivent concentrer leurs efforts et activer ces technologies d'atténuations sur leurs produits. Dans le cadre de ce programme, Microsoft propose de guider les distributeurs de logiciels, afin de leur permettre d'activer différentes technologies d'atténuation d'exploitations dans leurs produits :
http://msdn.microsoft.com/en-us/library/bb430720.aspx.

Conclusion

Il est impossible d'empêcher complètement l'introduction de vulnérabilités lors du développement de projets logiciels à grande échelle. La tendance séculaire dénote que des milliers de vulnérabilités de sécurité sont, chaque année, divulguées de part en part dans le secteur logiciel. Ces vulnérabilités sont pour la plupart de haute sévérité dans les applications et relativement faciles à exploiter.

Microsoft a fait des progrès en termes d'utilisation de SDL et de la science de la sécurité, en vue de réduire les vulnérabilités et d'intégrer les atténuations de menace au sein des logiciels et services Microsoft, de façon à mieux protéger les clients Microsoft et les internautes. Les résultats de recherches reportés ici démontrent que si un grand nombre des applications les plus connues mondialement tirent parti des atténuations de sécurité intégrées aux systèmes d'exploitation Windows, les opportunités d'amélioration n'ont pas fini de se présenter.

Les distributeurs de logiciels et les entreprises de développement de logiciels indépendants qui n'utilisent pas encore SDL se doivent de considérer sa mise en application, afin de se rendre compte des avantages qu'il apporte.

Pour en savoir plus sur le processus SDL, son utilisation dans les entreprises de développement et les avantages liés au développement en toute sécurité, consultez la page suivante :
http://www.microsoft.com/sdl

Pour en avoir plus sur la science de la sécurité chez Microsoft, consultez la page suivante :
http://www.microsoft.com/msec

Microsoft

One Microsoft Way
Redmond, WA 98052-6399
microsoft.com/security