



# Informe de progreso del proceso SDL

Avances en la reducción de las vulnerabilidades del software  
y el desarrollo de la mitigación de amenazas en Microsoft

2004 - 2010

**Microsoft®**



## Informe de progreso del proceso SDL

©2011 Microsoft Corporation. Todos los derechos reservados. Este documento se proporciona “tal cual”. La información y opiniones expresadas en este documento, incluida cualquier URL y otras referencias a sitios Web de Internet, pueden cambiar sin previo aviso. Usted acepta el riesgo que pueda conllevar su utilización. Este documento no le proporciona ningún derecho legal en relación a ninguna propiedad intelectual respecto de cualquier producto de Microsoft. Podrá copiar y utilizar este documento, a efectos de consulta internos.

## Autores


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*

# Prólogo

---

Este año he cumplido un importante hito en mi carrera profesional: hace 40 años escribí mi primer informe sobre el tema de la seguridad del software. Al reflexionar sobre este hito, tres cosas me pasaron por la mente: En primer lugar, ¡que hace mucho que trabajo en esta actividad! En segundo lugar, que muchos de los enfoques para compilar software seguro simplemente no funcionaron. En tercer lugar, creo que en este momento la industria ha desarrollado algunos enfoques eficaces para compilar software más seguro, y siento un optimismo cauteloso con respecto al futuro.

He pasado la mayor parte de los últimos once años trabajando en el Trustworthy Computing Group en Microsoft, implementando principios de seguridad y privacidad en los procesos de desarrollo de software y la cultura de la empresa. Una parte fundamental de Trustworthy Computing es el ciclo de vida de desarrollo de seguridad (SDL, Security Development Lifecycle). SDL es un proceso de control de seguridad dedicado al desarrollo de software y la introducción de seguridad y privacidad en todas las fases del proceso de desarrollo. SDL ha sido una directiva obligatoria para toda la empresa desde 2004. Combina un enfoque holístico y práctico destinado a reducir el número y la severidad de las vulnerabilidades de los productos y servicios de Microsoft, y de esta manera, limitar las oportunidades para que los atacantes puedan poner en riesgo los equipos. Compartimos libremente el proceso SDL con las organizaciones de desarrollo y la industria del software, y nos complace ver cómo ha sido adoptado (a veces en formatos adaptados) por una serie de proveedores de software y hardware independientes, autoridades gubernamentales y organizaciones de desarrollo de usuarios finales.



Incluso antes de que se formalizara el proceso SDL, organizamos un equipo pequeño para investigar las vulnerabilidades de seguridad, sus causas, y formas sistemáticas de eliminarlas o de mitigar sus efectos. Con el tiempo nos empezamos a referir a este equipo y sus actividades como “ciencia de la seguridad”. La ciencia de la seguridad es la “ciencia dentro del proceso SDL”. Estudiamos la forma en que se ataca a los sistemas informáticos, y cómo se pueden contrarrestar estos ataques, y desarrollamos herramientas y técnicas avanzadas que ayudan a dificultar que se concreten los ataques contra el software. Cuando estamos convencidos de que estas herramientas y técnicas resultan confiables y eficaces, exigimos su aplicación como parte del proceso SDL. A continuación, las damos a conocer al público, de manera que nuestros clientes, socios e incluso nuestros competidores puedan compilar software más seguro.

En este informe, conocerá la evolución del proceso SDL y el progreso que hemos logrado en el uso del proceso SDL y la ciencia de la seguridad para reducir las vulnerabilidades y mitigar las amenazas contra el software y los servicios de Microsoft. Creemos que el proceso SDL nos ha ayudado a proteger a los clientes de Microsoft y, gracias a su amplio nivel de adopción, creemos que también ha ayudado a proteger a la comunidad de usuarios de Internet en general.

Si usted es un proveedor de software independiente u otro desarrollador de software y ya está usando el proceso SDL, este informe le permitirá conocer algunos de los antecedentes del proceso SDL y la forma en que ha madurado en los últimos seis años. Si todavía no está usando el proceso SDL, esperamos que este informe le ayude a comprender por qué creemos que es un proceso eficaz y eficiente, y lo alentamos a que lo pruebe en su propia organización.

Steve Lipner  
Director Principal de Estrategia de Ingeniería de Seguridad  
Trustworthy Computing Security, Microsoft

# Introducción

---

Las vulnerabilidades son puntos débiles en el software que permiten que un atacante ponga en riesgo la integridad, disponibilidad o confidencialidad de ese software o de los datos que procesa. Algunas de las vulnerabilidades más graves permiten que los atacantes ejecuten un código de software de su elección, poniendo potencialmente en riesgo al equipo, su software y los datos que residen en ese equipo. La divulgación de una vulnerabilidad puede provenir de diversas fuentes, entre ellas los proveedores de software, proveedores de software de seguridad, investigadores de seguridad independientes y los creadores de software malintencionado (también conocido como “malware”).

Es imposible evitar por completo que se introduzcan vulnerabilidades durante el desarrollo de proyectos de software a gran escala. Mientras los seres humanos escriban código de software, siempre se cometerán errores que pueden provocar imperfecciones en el software: ningún software es perfecto. Algunas imperfecciones o errores simplemente evitan que el software funcione exactamente de la manera deseada, pero otros errores pueden causar vulnerabilidades. No todas las vulnerabilidades son iguales: hay algunas que los atacantes no pueden explotar porque mitigaciones específicas impiden que esos atacantes las usen. No obstante, es posible que un cierto porcentaje de las vulnerabilidades que existen en un software dado puedan explotarse.

Los datos de la National Vulnerability Database<sup>1</sup> demuestran que la tendencia a largo plazo para las divulgaciones de vulnerabilidades en toda la industria se caracteriza por miles de divulgaciones de vulnerabilidades todos los años, la mayoría de las cuales son de alta gravedad y baja complejidad. Además, la mayoría de las divulgaciones de vulnerabilidades se producen en las aplicaciones, en lugar de en los sistemas operativos o exploradores web. Esta tendencia es preocupante porque esencialmente significa que hay miles de divulgaciones de vulnerabilidades de alta gravedad en aplicaciones, y la mayoría de ellas son relativamente fáciles de explotar.

---

<sup>1</sup> La información de esta sección se ha compilado a partir de los datos de divulgaciones de vulnerabilidades que se publican en la National Vulnerability Database (<http://nvd.nist.gov>), el repositorio del gobierno de EE.UU. sobre datos de administración de vulnerabilidades basada en normas representados a través del protocolo de automatización de contenido de seguridad (SCAP).

Figura 1: *izquierda*: Divulgaciones de vulnerabilidades por gravedad en toda la industria, 2006 – 2010; *derecha*: Vulnerabilidades en toda la industria, por complejidad de acceso, 2006 – 2010

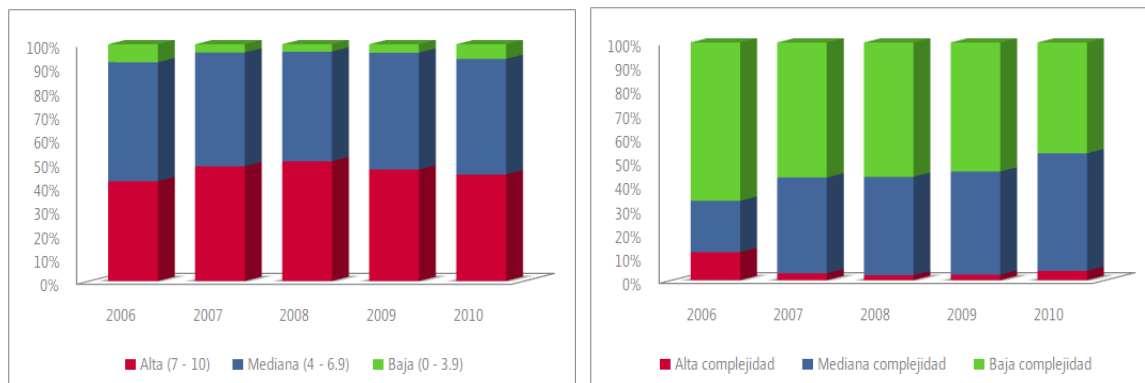
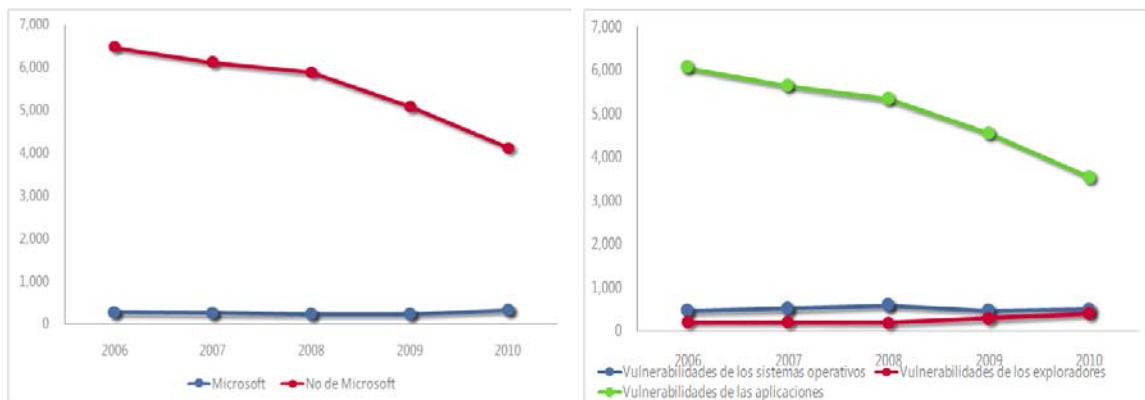


Figura 2: *izquierda*: Divulgaciones de vulnerabilidad para productos de Microsoft y de otras empresas, 2006 – 2010; *derecha*: Vulnerabilidades de sistemas operativos, exploradores y aplicaciones en toda la industria, 2006 – 2010



Microsoft considera que la información confidencial y personal debe protegerse, que las empresas de software deben adherir a prácticas comerciales que promuevan la confianza, y que la industria de la tecnología debe concentrarse en la ingeniería sólida y los procedimientos recomendados para ofrecer productos y servicios confiables y seguros. Nuestro enfoque para encarar estos desafíos se denomina Trustworthy Computing, o informática de confianza: un proyecto de colaboración a largo plazo para crear y ofrecer experiencias informáticas seguras, privadas y confiables para todos.

Una parte fundamental de Trustworthy Computing es el ciclo de vida de desarrollo de seguridad (SDL, Security Development Lifecycle). SDL es un proceso de control de seguridad dedicado al desarrollo de software y la introducción de seguridad y privacidad en todas las fases del proceso de desarrollo. Como directiva obligatoria para toda la empresa desde 2004, el proceso SDL ha tenido un papel fundamental en la incorporación de seguridad y privacidad en el software y la cultura de Microsoft. El proceso SDL combina un enfoque holístico y práctico para reducir el número y la severidad de las vulnerabilidades de los productos y servicios de Microsoft,

limitando de esta manera las oportunidades para que los atacantes pongan en riesgo los equipos. Microsoft comparte libremente el proceso SDL con las organizaciones de desarrollo de los clientes y la industria del software, donde se lo ha usado para desarrollar software más seguro.

Figura 3: Procedimientos que definen el proceso de ciclo de vida de desarrollo de seguridad (SDL)



La ciencia de la seguridad es la ciencia dentro del proceso SDL, que se desarrolla sobre una base innovadora de investigación para entender cómo los sistemas informáticos pueden sufrir ataques y de qué manera estos ataques pueden prevenirse o mitigarse. La ciencia de la seguridad entonces incuba y desarrolla herramientas y técnicas de última tecnología que ayudan a dificultar la concreción de los ataques al software. La ciencia de la seguridad mejora el proceso SDL de tres maneras:

- Ayuda a detectar vulnerabilidades del software.
- Desarrolla técnicas y herramientas de mitigación de vulnerabilidades de seguridad que los desarrolladores deberían adoptar.
- Supervisa constantemente las tendencias y la actividad en el entorno de las amenazas y mejora las herramientas y procesos sobre la base de esas observaciones. Si las tareas de supervisión determinan que ha surgido una nueva amenaza en el ecosistema, intervienen los procesos de respuesta de seguridad de Microsoft.



## Beneficios del desarrollo seguro

Las organizaciones que inician programas de desarrollo de software por lo general se concentran en la funcionalidad en primer lugar y agregan las pruebas de seguridad al final del proceso de desarrollo. Este enfoque, sin embargo, tiene desventajas significativas en comparación con un enfoque estructurado que integra las tareas de seguridad a lo largo de todo el proceso de desarrollo.

El National Institute of Standards and Technology (NIST) estima <sup>2</sup> que las correcciones al código efectuadas después del lanzamiento tienen un costo 30 veces superior al costo de las correcciones realizadas durante la fase de diseño. Recientes estudios de Forrester Research, Inc. y Aberdeen Group han concluido que, cuando las empresas adoptan un proceso estructurado como Microsoft SDL, que sistemáticamente se ocupa de la seguridad del software durante la fase correspondiente del ciclo de vida, existe una probabilidad mucho mayor de detectar y corregir las vulnerabilidades en las primeras etapas del ciclo de desarrollo, reduciendo de esta manera el costo total del desarrollo de software.

En enero de 2011, Forrester publicó <sup>3</sup> los resultados de una encuesta patrocinada por Microsoft entre las personas con mayor influencia en el desarrollo de software en toda América del Norte. Forrester concluyó que, aunque la seguridad de las aplicaciones no era una práctica madura para muchos, quienes aplicaban un método preceptivo y coordinado tenían una mejor rentabilidad de la inversión.


En agosto de 2010, Aberdeen Group publicó <sup>4</sup> una investigación que confirmaba que el costo anual total de las iniciativas de seguridad de las aplicaciones se ve ampliamente compensado por los beneficios acumulados. En diciembre de 2010, Aberdeen Group publicó las conclusiones de investigaciones más detalladas de organizaciones que implementaban programas estructurados para el desarrollo de seguridad y "... llegó a la conclusión de que obtenían una rentabilidad muy sólida de 4,0 veces sobre sus inversiones anuales en la seguridad de las aplicaciones".

---

<sup>2</sup> Informe de NIST de 2002; [Los impactos económicos de la infraestructura inadecuada de las pruebas de software](#)

<sup>3</sup> 2011 Forrester Consulting; [Estado de la seguridad de las aplicaciones: Las prácticas inmaduras alimentan las ineficacias pero se puede alcanzar una rentabilidad de la inversión positiva](#)

<sup>4</sup> 2010 Aberdeen Group; [Protección de sus aplicaciones: tres formas de jugar](#)



Con esta información de fondo en mente, el resto de este informe estará dedicado al progreso que ha logrado Microsoft en el uso del proceso SDL y la ciencia de la seguridad para reducir las vulnerabilidades y desarrollar mitigaciones de las amenazas en el software y los servicios de Microsoft de maneras que ayuden a proteger a los clientes de Microsoft y a los usuarios de Internet. Se incluyen los resultados de la nueva investigación sobre algunas de las aplicaciones más populares del mundo para saber cuántas de estas aplicaciones realmente aprovechan las mitigaciones de seguridad que se incorporan en los sistemas operativos Windows®.

# Security Development Lifecycle

---

## Prehistoria de la seguridad en Microsoft (1990 - 2004)

Microsoft tiene una larga historia de implementación de seguridad de software, con algunos proyectos, como la compatibilidad con criterios comunes, que abarcan décadas. Sin embargo, antes de la adopción del proceso SDL en Microsoft, los procesos de seguridad se aplicaban de modo dispar: los equipos de los productos implementaban protecciones a la seguridad y privacidad en gran medida siguiendo su propio criterio. Algunos equipos de desarrollo sometían sus aplicaciones a un examen exhaustivo, mientras que otros daban prioridad a las nuevas funciones y la funcionalidad por encima de la seguridad y privacidad.

### Amenazas emergentes y la respuesta de Microsoft

Una serie de incidentes de alto perfil relacionados con software malintencionado entre mediados y fin de la década de 1990 (Melissa) y principios de la década de 2000 (Code Red, Nimda, UPnP, etc.) llevaron a Microsoft a reformular su proceso y estrategia de seguridad para los desarrolladores. Los primeros elementos de lo que se transformó en el **proceso de modelado de amenazas** (por ejemplo, [STRIDE](#)) se introdujeron durante este período, junto con el concepto de un **límite de errores** universal que establece la gravedad de la vulnerabilidad. El **análisis de causa raíz** formalizado se implementó para entender la causa y el impacto de diversos tipos de vulnerabilidad en todo el espectro de los productos de Microsoft. Se introdujeron otros cambios basados en la tecnología, entre ellos el **análisis de código estático**, cuando Microsoft adquirió Intrinsic y su herramienta de análisis estático PREfix. Muchas de las primeras tareas de seguridad se aplicaron de forma limitada a Windows 2000 y con el tiempo se convirtieron en el sustento de base del proceso SDL como se practica hoy.

STRIDE es la taxonomía usada en Microsoft para clasificar las amenazas detectadas durante las actividades de modelado de amenazas.

- *Suplantación de identidad*
- *Manipulación*
- *Rechazo*
- *Divulgación de información*
- *Denegación de servicio*
- *Elevación de privilegios*

Aunque estos cambios en realidad tuvieron un impacto positivo sobre el software de Microsoft, no eran obligatorios ni lo suficientemente completos como para enfrentar los problemas que presentaban los productos de Microsoft. Las interrupciones causadas por la aparición de software malintencionado que explotaba las vulnerabilidades del software de Microsoft afectaron a muchos clientes de Microsoft y usuarios de Internet, lo que provocó una erosión de la confianza. Como resultado, Bill Gates publicó su memorando sobre Trustworthy Computing en enero de 2002. El lanzamiento de Trustworthy Computing representó un cambio fundamental en las prioridades de la empresa en lo que se refiere a la seguridad del software. Este mandato ejecutivo posicionaba la seguridad como prioridad principal para Microsoft y ofrecía el impulso necesario para una campaña permanente de cambios en la cultura de ingeniería.

### NET y los movimientos de seguridad de Windows

El apoyo ejecutivo al más alto nivel ofreció a principios de 2002 a los expertos en seguridad de Microsoft una oportunidad “autorizada” para probar la aplicación aplicando un enfoque más holístico para el desarrollo seguro. Microsoft detuvo temporalmente el desarrollo de .NET Framework Common Language Runtime (CLR) para cambiar el enfoque del equipo de desarrollo, dando prioridad a la escritura de código seguro en lugar a de las funciones y la funcionalidad. Durante unas diez semanas, se detectaron y corrigieron decenas de errores de seguridad, y el énfasis colectivo en la seguridad introdujo nuevos métodos para proteger el software, con la **reducción de la superficie expuesta a los ataques** como ejemplo notable.

El análisis y reducción de la superficie expuesta a ataques es una actividad de análisis de la seguridad que se ejecuta durante la fase de diseño de Microsoft SDL, y tiene como fin la reducción de la cantidad de código y datos expuestos a los usuarios no confiables.

El trabajo en .NET CLR se consideró como un éxito, y como resultado, los ejecutivos de administración aceptaron que debía aplicarse un enfoque similar en lo que en ese momento era el lanzamiento inminente de Windows Server 2003. Esta tarea pasó a ser conocida como “movimiento de seguridad de Windows” (Windows Security Push). La aplicación de estos procesos a las tareas de desarrollo de Windows presentó una serie de desafíos logísticos y tecnológicos desalentadores: por ejemplo, el tamaño del código base de Windows (en ese momento) era aproximadamente diez veces mayor que el de CLR. El tamaño y complejidad de organización de la División Windows también promovía un aumento de la confianza en la tecnología, pero las innovaciones en los procesos siguieron apareciendo. *El modelado de amenazas con herramientas* estaba en su infancia, y además de la solución PREFIX existente, se presentó el *análisis estático ligero* (PREfast) en los escritorios de los desarrolladores como una manera de encontrar y corregir errores antes de la protección del código. Además, se agregó el requisito de una “*auditoría de seguridad*”: esta auditoría fue precursora de la *Revisión Final de Seguridad (FSR, Final Security Review)*, que desde entonces ha pasado a ser un elemento fundamental del proceso SDL en Microsoft.

.NET CLR, Windows Server 2003, SQL Server 2000 SP3, y otros movimientos de seguridad produjeron resultados impactantes. Sin embargo, los líderes de seguridad de Microsoft entendieron que un “movimiento” antes del lanzamiento del producto no sería tan eficaz como la integración de la seguridad en el diseño y desarrollo de los productos. Con una confianza respaldada por la experiencia de los movimientos de seguridad, se presentó una propuesta de proceso obligatorio a principios de 2004 ante el Equipo de Dirección Principal de Microsoft: la propuesta fue aprobada como directiva de Microsoft, con el resultado de que todos los productos y servicios en línea que estaban expuestos a un riesgo de seguridad significativo o la información confidencial de ese proceso debían estar de acuerdo con los requisitos del proceso SDL.

## Microsoft SDL (2004 – Actualidad)

Con el proceso SDL, Microsoft ha elegido un enfoque incremental para el perfeccionamiento de los procesos de seguridad que agrega mejoras a la seguridad y privacidad de los procesos y la tecnología a medida que maduran. Como es de esperarse de un proceso que madura con el tiempo, muchas técnicas y procesos de seguridad se agregaron en las primeras etapas de la evolución del proceso SDL. A medida que pasó el tiempo, el enfoque acumulativo se ha trasladado a la eficiencia y productividad de las herramientas y procesos de SDL, y se han agregado mejoras como requisitos o recomendaciones. Algunas adiciones al proceso SDL se ocupan de nuevas amenazas, mientras que otras son mejoras de los requisitos existentes.

Los grupos de productos de Microsoft usaban varias técnicas y herramientas de seguridad antes de su inclusión como requisitos o recomendaciones de seguridad de Microsoft SDL

Los siguientes puntos que se resaltan en **negrita no** son una lista exhaustiva año por año de todas las adiciones al proceso SDL, sino que son adiciones importantes al proceso. Se puede consultar una discusión detallada de los requisitos de Microsoft SDL (desde la versión 3.2) en el [centro de desarrolladores de MSDN](#). Además, estos puntos reflejan el momento en que los procesos o tecnologías se convierten en **requisitos** de SDL. Sin embargo, es importante observar que varias técnicas eran usadas por los grupos de productos de Microsoft, o se incluyeron en el proceso SDL como recomendaciones, mucho antes de convertirse en requisitos de SDL. Por motivos de brevedad y claridad, se han omitido las escalas de tiempo para las recomendaciones del proceso SDL.

### 2004 – SDL 2.0

La primera versión oficial de Microsoft SDL, creada en 2004, era fundamentalmente una lista codificada de perfeccionamientos de las técnicas y procesos que se habían usado anteriormente durante las tareas de NET y los movimientos de seguridad de Windows que se describieron anteriormente. Algunos elementos figuraban como requisitos y otros se incluyeron como recomendaciones. La versión inicial del proceso SDL garantizaba que las acciones como el modelado de amenazas, análisis estático y la revisión final de seguridad fueran obligatorias para el software de Microsoft que estuviera expuesto a un riesgo de seguridad significativo.

El concepto de “**Riesgo de seguridad significativo**” se usa para determinar cuáles de las aplicaciones están sujetas a las restricciones de Microsoft SDL. En resumen, cualquier aplicación que presente una o más de las siguientes características debe estar de acuerdo con el proceso SDL:

- Se encuentra ampliamente implantado en una empresa u organización
- Procesa información confidencial, lo que incluye información de identificación personal (PII)
- “Escucha” o interactúa en una red

## 2005 – SDL 2.1 y 2.2

**Límite de errores.** El límite de errores (bug bar) es un criterio de calidad que se aplica a un proyecto completo de desarrollo de software. Se agregó al proceso SDL como una manera de establecer los criterios de calidad de la versión según qué tan críticos fueran los errores. Se define al principio de un proyecto para mejorar la comprensión de los riesgos asociados con problemas de seguridad y permite que los equipos identifiquen y corrijan errores de seguridad durante el desarrollo. El equipo de proyecto negocia un límite de errores con su asesor de seguridad asignado, con aclaraciones específicas para el proyecto y (según corresponda) requisitos de seguridad más estrictos estipulados por el asesor de seguridad. Una vez definido, el límite de errores nunca se flexibiliza.

**Exploración de vulnerabilidades (archivo y llamada a procedimiento remoto (RPC)).** Se empezó a aplicar la introducción deliberada de datos aleatorios o no válidos en un programa (lo que se denomina “exploración de vulnerabilidades”) para ayudar a detectar errores de aserción, pérdidas de memoria y bloqueos. Ésta era una técnica exitosa usada por el ecosistema de los atacantes, y debido a su eficacia, Microsoft la adoptó como una práctica de prueba obligatoria. Las técnicas de exploración de vulnerabilidades se concentraron inicialmente en analizadores de archivos e interfaces RPC.

**Normas criptográficas.** Se implantó una serie de requisitos en torno al uso de criptografía en las aplicaciones de Microsoft para exigir que los equipos de productos usen normas criptográficas establecidas en lugar de intentar desarrollar sus propios algoritmos no estándar. Otros requisitos estipulan el uso de bibliotecas criptográficas (como CryptoAPI), desalientan el uso de soluciones codificadas de forma rígida (criptografía personal), y especifican el uso de algoritmos criptográficos seguros con notificación deliberada a los usuarios si es necesario recurrir a un algoritmo menos seguro por motivos de compatibilidad.

**Comprobación en tiempo de ejecución.** Además de la exploración de vulnerabilidades, el proceso SDL exigía pruebas adicionales de los programas en tiempo de ejecución. Mediante el uso de AppVerifier y otras herramientas, los equipos detectaron errores en una aplicación de destino mediante la supervisión pasiva y los informes sobre el comportamiento del programa mientras se ejecutaba en su entorno operativo planificado.

## 2006 – SDL 3.0 y 3.1

**Exploración de vulnerabilidades (ActiveX).** La exploración de vulnerabilidades se amplió a los controles ActiveX® incluidos en las aplicaciones de Microsoft. Los controles ActiveX se pueden usar como una forma de inyectar datos no confiables desde sitios web y otros lugares en un equipo host. Una saturación del búfer en un control ActiveX por script puede llevar a la ejecución de código desde la zona de Internet en el contexto de un usuario registrado.

### **Interfaces de programación de aplicaciones (API) vedadas (Banned.h).**

La biblioteca en tiempo de ejecución C se creó hace más de 25 años, cuando la conectividad de red y el entorno de amenazas era mucho menos problemático. Microsoft decidió dejar de utilizar un subconjunto de la biblioteca en tiempo de ejecución C para eliminar funciones que, según se detectó, presentaban amenazas de seguridad, especialmente saturaciones del búfer. Al principio este requisito se articuló simplemente bajo la forma de una lista de API defectuosas, pero con el tiempo se transformó en un archivo de encabezado (Banned.h) que se podía usar en conjunción con un compilador para ayudar a proporcionar un método automatizado para corregir el código fuente.

**Normas de privacidad para el desarrollo.** Microsoft estableció amplias directivas internas para los desarrolladores que se concentraban en la protección de la privacidad de clientes y usuarios. Estas directivas ayudaron a los desarrolladores a entender las expectativas de los usuarios, las leyes globales de privacidad y los procedimientos aprobados para garantizar la privacidad en los productos y servicios de Microsoft.

**Requisitos de servicios en línea.** Antes de SDL 3.1, había dos conjuntos diferenciados de requisitos de seguridad: el proceso SDL para el desarrollo de aplicaciones cliente/servidor (aplicado a los equipos de productos tradicionales) y un conjunto diferente de requisitos de seguridad que se aplicaban al desarrollo de MSN y otros servicios en línea. El requisito de seguridad en los servicios en línea se agregó para unificar los elementos de los dos procesos en un proceso SDL coherente.



## 2007 – SDL 3.2

**Defensas de scripts de sitios.** Los procedimientos obligatorios del proceso SDL para mitigar o prevenir el uso de ataques de scripts de sitios (XSS), entre ellos validación de entrada, codificación de salida y exploración de vulnerabilidades de caja negra. Este requisito también estipuló el uso de la biblioteca Anti-XSS para la codificación de salida, que usa el principio de inclusión (también conocido como “listas blancas”).

**Defensas contra ataques por inyección de código SQL.** Microsoft incorporó orientación específica de SDL en los ataques por inyección de código SQL, lo que incluye el uso de procedimientos almacenados y consultas con parámetros, además de exploraciones de seguridad de caja negra.

**Defensas de análisis de XML.** Se agregó un requisito para enfrentar los ataques de análisis de XML. Debido a que un servicio web por lo general intenta analizar cualquier XML válido que se le transfiere, es fundamental hacer una exploración de vulnerabilidades de todas las interfaces para garantizar la validación de entrada correcta. Además de explorar las vulnerabilidades del analizador, se incluyeron requisitos para versiones seguras específicas de analizadores de XML.

**Primer lanzamiento público del proceso Microsoft SDL.** En respuesta a consultas de los clientes, usuarios y otros interesados, Microsoft publicó el proceso SDL para aumentar la transparencia y permitir una mejor comprensión de los procesos y tecnologías usados para proteger el software de Microsoft.

## 2008 – SDL 4.0 y 4.1

**Selección aleatoria del diseño de espacio de direcciones (ASLR).** La selección aleatoria del diseño de espacio de direcciones, que era una recomendación, pasó a ser un requisito. Este requisito especificaba que se habilitara ASLR en todos los archivos binarios de código nativo (C/C++) para protegerse contra los ataques de *volver a biblioteca libc (return-to-libc)*. La ASLR se describe en profundidad en la sección de “Mitigaciones de la ciencia de la seguridad” de este documento.

**CAT.NET para servicios en línea/de código administrado.** Se especificó el uso de CAT.NET para garantizar que se realice el análisis de código estático para código administrado (.NET/C#) antes de la protección del código.

**Defensas contra falsificación de solicitudes entre sitios (CSRF).** Se agregó un requisito (ViewStateUserKey) para garantizar que se usen tokens únicos por sesión aleatorios para prevenir los ataques de CSRF contra las aplicaciones de web implementadas en lenguajes de .NET.

## 2009 – SDL 5.0

**Exploración de vulnerabilidades (red).** Requisitos ampliados de exploración de vulnerabilidades para todos los analizadores e interfaces de red. Aumento de los umbrales de aceptabilidad para las exploraciones de vulnerabilidades (100.000 iteraciones correctas).

**Revisiones de seguridad operativa.** Todas las aplicaciones destinadas a ejecutarse en centros de datos de Microsoft deben pasar por una revisión de seguridad operativa adicional antes de implantarse, además de cumplir todos los criterios de desarrollo de seguridad del proceso SDL. Si bien el requisito de las revisiones de seguridad operativa ya existía antes, se integró en el proceso SDL para garantizar que los equipos de servicios en línea tuvieran un marco unificado de requisitos de seguridad.

**Requisitos de seguridad para licencias de terceros.** Se ampliaron los procedimientos existentes de desarrollo de SDL y los criterios de servicios de seguridad a *todo* el código de terceros con licencia para usarse en productos y servicios de Microsoft.

**Lanzamiento de herramientas externas.** Primer lanzamiento público de herramientas del equipo Microsoft SDL.

- [SDL Threat Modeling Tool](#) permite que los expertos en temas ajenos a la seguridad creen y analicen modelos de amenazas.
- [SDL Template for Visual Studio® Team System \(Classic\)](#), una plantilla que integra automáticamente la directiva, el proceso y las herramientas asociados con la guía de procesos de Microsoft SDL versión 4.1 directamente en el entorno de desarrollo de software de Visual Studio Team System.
- [SDL Binscope Binary Analyzer](#), una herramienta de comprobación que analiza archivos binarios para garantizar que se hayan desarrollado conforme a los requisitos y recomendaciones del proceso SDL.
- [SDL Minifuzz File Fuzzer](#), una herramienta básica de exploración de vulnerabilidades diseñada para ayudar a detectar problemas que pueden exponer vulnerabilidades de seguridad en el código de administración de archivos.

## 2010 – SDL 5.1

**Compatibilidad del “código muestra” con SDL.** El código muestra se usa como plantilla para varios proyectos de desarrollo. Los errores de seguridad en el código muestra a menudo significan errores de seguridad en un código desarrollado por terceros que aprovecha el código muestra. Todo el código muestra enviado con los productos de Microsoft debe cumplir los mismos requisitos de seguridad que nuestros productos.

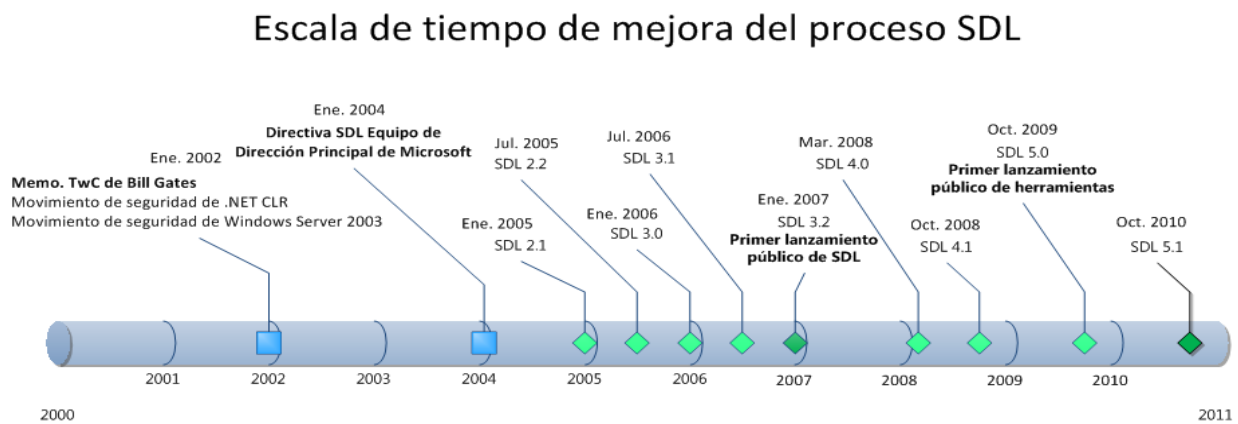
### **Lanzamiento público de la implementación simplificada de Microsoft SDL**

Microsoft SDL se basa en conceptos de seguridad comprobados adaptados para satisfacer las necesidades comerciales y técnicas específicas de Microsoft. La [implementación simplificada de Microsoft SDL](#) es una adaptación simplificada no protegida por derechos de propiedad de Microsoft SDL, adecuada para su uso en otros entornos y escenarios de desarrollo y en otras plataformas de software.

**Lanzamiento de herramientas externas.** Segundo lanzamiento público de herramientas del equipo de Microsoft SDL.

- [SDL MSF+Agile Template for Visual Studio Team System](#), una plantilla de Team Foundation Server que incorpora automáticamente la directiva, los procesos y las herramientas asociados con SDL para la guía de Agile Development en la plantilla de proceso conocida de Microsoft Solutions Framework (MSF) para Agile Software Development (MSF-Agile) que se envía con Visual Studio Team System.
- [Regular Expressions \(Regex\) Fuzzer](#), una herramienta de comprobación diseñada para ayudar a probar expresiones regulares de posibles vulnerabilidades de denegación de servicio.

Figura 4: Escala de tiempo de hitos importantes en la evolución del proceso SDL en Microsoft



Microsoft ha estado usando el proceso SDL o sus precursores durante casi una década para inyectar procedimientos comprobados de seguridad en el desarrollo de nuestro software. Aunque puede resultar tentador considerar que el proceso SDL y las metodologías de desarrollo de seguridad están “completos”, la realidad es que las amenazas al software no son estáticas ni nunca lo serán. Como resultado, debemos seguir supervisando el entorno de las amenazas, hacer las inversiones necesarias en personal y tecnología para perfeccionar y mejorar el proceso Microsoft SDL, y compartir de forma dinámica los procedimientos recomendados con otros desarrolladores de software para ayudar a crear una experiencia informática más segura para todos.

# La ciencia de la seguridad

---

## Las mitigaciones ayudan a administrar el riesgo

Es un escenario desafortunado pero realista: se ha divulgado públicamente una vulnerabilidad en uno de sus productos y todavía no hay ninguna revisión disponible. ¿Cómo puede ayudar a que sus clientes se protejan a sí mismos? Como desarrollador de software, es fundamental que la respuesta a esta pregunta se entienda bien antes de que se plantee una situación de este tipo. El reflexionar acerca de este problema con anticipación hace que sea posible incorporar funciones de defensa profundas en su software, que posteriormente pueden permitir que los clientes administren el riesgo con mayor eficacia al enfrentarse a una vulnerabilidad desconocida o para la que no existe una revisión. Estas funciones de defensa profundas generalmente se denominan *mitigaciones* porque pueden mitigar de forma parcial o completa el riesgo asociado con una vulnerabilidad.

A menudo, se pueden asumir distintos enfoques para mitigar una vulnerabilidad. Por ejemplo, una vulnerabilidad en un servicio de red se puede mitigar evitando la conectividad (firewall), evitando el acceso (autenticación/autorización), deshabilitando el servicio o la función vulnerable (configuración), limitando la capacidad del atacante (contención), y así sucesivamente. En cada caso, el objetivo es el mismo: hacer que sea imposible o muy costoso que un atacante explote con éxito una vulnerabilidad. Por lo tanto, las mitigaciones que permiten que este objetivo se concrete con éxito pueden mantener protegidos a los clientes mientras se desarrolla e implanta una actualización de seguridad.

Un enfoque particularmente valioso para concretar este objetivo es concentrarse en contrarrestar las *técnicas de explotación* que aplican los atacantes cuando desarrollan una forma de explotar una vulnerabilidad. Se puede considerar a las técnicas de explotación como las herramientas que han desarrollado los atacantes para convertir una vulnerabilidad en algo que le permite al atacante ejecutar un código arbitrario y asumir el control del equipo de un usuario. Al contrarrestar o desestabilizar estas técnicas esencialmente se elimina una herramienta de la caja de herramientas del atacante y (en el mejor de los casos) se puede hacer que sea imposible explotar la vulnerabilidad o aumentar el tiempo y el costo de desarrollo de una técnica de explotación. Estas tareas tienen un impacto directo sobre el incentivo económico de un atacante para explotar una vulnerabilidad y, al mismo tiempo, protegen a los clientes mientras se desarrolla e implanta una revisión. Las mitigaciones que adoptan este enfoque generalmente se denominan *mitigaciones de vulnerabilidades de seguridad*.

Las mitigaciones de vulnerabilidades de seguridad presentan algunas características adicionales que las hacen bastante atractivas como estrategia de mitigación. En muchos casos, la lógica que se necesita para contrarrestar una técnica de explotación se puede incorporar en una aplicación. El uso de un enfoque de este tipo significa que los clientes no están obligados a adaptar una mitigación en su entorno existente cuando se detecta una nueva vulnerabilidad. Otro beneficio es que, dado que las mitigaciones de vulnerabilidades de seguridad se centran en contrarrestar las técnicas de explotación, generalmente son transparentes a la funcionalidad de las aplicaciones típicas. Si son transparentes, las mitigaciones de vulnerabilidades de seguridad se pueden habilitar de forma predeterminada sin afectar la funcionalidad de la aplicación. Incluir las mitigaciones de vulnerabilidades de seguridad en las aplicaciones y habilitarlas de forma predeterminada permite ofrecer protección genérica para vulnerabilidades que son conocidas o que actualmente sean desconocidas.

Debido a estos beneficios, no debería ser ninguna sorpresa que Microsoft haya usado la ciencia de la seguridad para desarrollar una amplia gama de tecnologías de mitigación de vulnerabilidades de seguridad durante la última década. Estas tecnologías ahora se integran en herramientas de desarrollo como Visual Studio y también se incorporan en los sistemas operativos Windows propiamente dichos. Este nivel de integración permite que Microsoft y otros desarrolladores de software compilen aplicaciones con mitigaciones que están incorporadas y habilitadas de forma predeterminada.

## Tácticas

Hasta ahora, en este documento sólo se han examinado las mitigaciones de vulnerabilidades de seguridad a un nivel alto. Esta sección explora algunas de las tecnologías específicas y tácticas fundamentales que se usan para contrarrestar las técnicas de explotación, para que pueda entender mejor cómo funcionan las mitigaciones de vulnerabilidades de seguridad. El conjunto de tácticas que usan las mitigaciones de vulnerabilidades de seguridad se puede agrupar en una serie de categorías distintas, entre ellas la aplicación de invariables, la adición de diversidad artificial y el aprovechamiento de los déficits de conocimiento. Las siguientes subsecciones ilustran la forma en que funcionan estas tácticas resaltando ejemplos específicos de tecnologías de mitigación de vulnerabilidades de seguridad que las utilizan. A menudo es común que las mitigaciones de vulnerabilidades de seguridad combinen una o más de estas tácticas para maximizar su eficacia.

### Aplicación de invariables

Una táctica que se puede usar para contrarrestar las técnicas de explotación es aplicar nuevos invariables que invaliden las suposiciones implícitas de una o más técnicas de explotación. Una analogía simple para esto puede ser el agregar barrotes a una ventana: mientras que anteriormente un atacante podía simplemente abrir la ventana y entrar, ahora tienen que eliminar los barrotes que impiden la entrada por la ventana. Esta idea simple se ha plasmado en forma de varias tecnologías de mitigación, dos de las más sobresalientes son *Prevención de ejecución de datos* (DEP) y *Protección contra sobrescritura del controlador de excepciones estructurado* (SEHOP).

### Prevención de ejecución de datos (DEP)

Una de las suposiciones que las vulnerabilidades de seguridad hacen a menudo es que los datos se pueden ejecutar como un código. El origen de esta suposición deriva del procedimiento común en que las vulnerabilidades de seguridad inyectan un código máquina personalizado (que a menudo se denomina *código shell*) y posteriormente lo ejecutan: el proceso conocido como ejecución de código arbitrario. En la mayoría de los casos, las vulnerabilidades de seguridad guardan ese código máquina personalizado en partes de la memoria de un programa, como la pila o el montón, que tradicionalmente están destinados a contener únicamente datos. Esta técnica de explotación históricamente ha sido bastante confiable porque los procesadores Intel más antiguos y las versiones de Windows anteriores a Windows XP Service Pack 2 no eran compatibles con la memoria no ejecutable. La introducción de DEP en Windows XP Service Pack 2 estableció un nuevo invariable que hizo posible evitar que los datos se ejecutaran como un código. Cuando se habilita DEP, ya no es posible que una vulnerabilidad de seguridad inyecte y ejecute directamente un código máquina personalizado desde regiones de memoria que están destinadas para datos.

## Protección contra sobrescritura del controlador de excepciones estructurado (SEHOP)

Determinados tipos de vulnerabilidades pueden permitir que un atacante use una técnica de explotación conocida como *Sobrescritura del controlador de excepciones estructurado* (sobrescritura de SEH). Esta técnica involucra dañar una estructura de datos que se usa cuando se administran condiciones excepcionales que se presentan mientras se ejecuta un programa. El daño a esta estructura de datos puede permitir que el atacante ejecute código desde cualquier parte de la memoria. Esta técnica se mitiga mediante SEHOP, que comprueba que la integridad de las estructuras de datos que se usan para administrar excepciones estén intactas. Este nuevo invariable hace que sea posible detectar el daño ocurrido cuando una vulnerabilidad de seguridad usa la técnica de sobrescritura de SEH y es en definitiva lo que hace posible contrarrestar las vulnerabilidades de seguridad que la usan. SEHOP es una tecnología de mitigación relativamente nueva y se espera que se convierta en un requisito en versiones futuras de SDL.

## Adición de diversidad artificial

La existencia de diversidad dentro de una población ayuda a minimizar el número de suposiciones universales que se pueden hacer acerca de los miembros de la población. Una analogía adecuada para esto se da a menudo en términos de biodiversidad: si surge un nuevo virus, la presencia de biodiversidad puede ayudar a garantizar que no toda la población se vea afectada. Este principio también se aplica en el mundo digital, donde los atacantes a menudo suponen que la configuración de un equipo será idéntica a la otro. La introducción de diversidad artificial en los equipos puede invalidar estas suposiciones y, por lo tanto, evitar que un atacante explote una vulnerabilidad de manera confiable. Un buen ejemplo de la adición de diversidad artificial se puede observar en el contexto de una mitigación de vulnerabilidades de seguridad conocida como selección aleatoria del diseño de espacio de direcciones (ASLR).



### Selección aleatoria del diseño de espacio de direcciones (ASLR)

Los atacantes a menudo dan por sentado que determinados objetos (como los archivos DLL) estarán ubicados en la misma dirección en la memoria cada vez que se ejecuta un programa (y en cada equipo en el que se ejecuta el programa). Las suposiciones de este tipo son convenientes para un atacante porque a menudo se requieren fundamentalmente para que la vulnerabilidad de seguridad tenga éxito. La imposibilidad de que un atacante codifique estas direcciones de forma rígida puede hacer que sea difícil o imposible escribir una vulnerabilidad de seguridad confiable que funcione en todos los equipos. Esta idea es la que impulsa la motivación por la ASLR. La ASLR puede contrarrestar varias técnicas de explotación introduciendo diversidad en el diseño de espacio de direcciones de un programa. En otras palabras, la ASLR realiza una selección aleatoria de la ubicación de los objetos en la memoria para evitar que un atacante pueda hacer suposiciones sobre su ubicación de manera confiable. Esta táctica tiene el efecto de hacer que el diseño de espacio de direcciones de un programa sea distinto en todos los equipos y es lo que en definitiva evita que un atacante haga suposiciones universales acerca de la ubicación de los objetos en la memoria.

### Aprovechamiento de los déficits de conocimiento

En algunas situaciones, las técnicas de explotación se pueden contrarrestar aprovechando secretos que el atacante desconoce o no puede predecir con facilidad. Una analogía simple para esta táctica puede ser una puerta con una cerradura con combinación. El enorme número de combinaciones posibles evita que el atacante pueda abrir sin dificultad la puerta, simplemente porque es poco viable que el atacante pueda adivinar la combinación a tiempo. El mismo concepto se aplica también para contrarrestar las técnicas de explotación. El uso de esta táctica en la práctica se demuestra más claramente con el soporte de la seguridad de generación de código (GS) que existe en el compilador de Microsoft Visual C++.

## GS

El ejemplo más conocido de una vulnerabilidad de software es la saturación del búfer basado en pilas. Este tipo de vulnerabilidad se explota tradicionalmente mediante la sobrescritura de datos críticos que se usa para ejecutar un código después de que se ha completado una función. Desde el lanzamiento de Visual Studio 2002, el compilador de Microsoft Visual C++ ha incluido soporte para el modificador del compilador /GS que, cuando está habilitado, introduce una comprobación de seguridad adicional que está diseñada para mitigar esta técnica de explotación. Esta táctica funciona colocando un valor aleatorio (conocido como cookie) antes de los datos críticos que un atacante podría querer sobrescribir. Luego se comprueba esta cookie cuando la función se completa para asegurarse de que sea igual al valor esperado. Si hay un error de coincidencia, se supone que se ha producido un daño y el programa puede terminar de manera segura. Este concepto simple demuestra cómo un valor secreto (en este caso la cookie) se puede usar para contrarrestar determinadas técnicas de explotación detectando daños en puntos críticos del programa. El hecho de que el valor sea secreto presenta un déficit de conocimiento que al atacante le resulta difícil resolver.

## Disponibilidad

Los diagramas de la Figura 5 y la Figura 6 resumen la disponibilidad de tecnología de mitigación de vulnerabilidades de seguridad de las versiones de sistemas operativos Windows a partir de Windows XP RTM<sup>5</sup> y Windows Server® 2003 RTM. Las mitigaciones de vulnerabilidades de seguridad marcadas como OptIn indican que la característica está deshabilitada de manera predeterminada y que las aplicaciones individuales deben habilitar explícitamente la característica (siendo OptOut lo inverso).

---

<sup>5</sup> RTM es una sigla que significa “listo para producción” que esencialmente significa que no se ha instalado ningún Service Pack.

Figura 5: Disponibilidad de mitigaciones de vulnerabilidades de seguridad en las SKU de cliente de Windows.

	XP RTM, SP1	XP SP2	XP SP3	Vista RTM	Vista SP1	Vista SP2	Win7 RTM
<b>SEH</b>							
SafeSEH	n	s	s	s	s	s	s
SEHOP	n	n	n	n	OptIn	OptIn	OptIn
Soporte OptIn para SEHOP por proceso	n	n	n	n	n	n	s
<b>Montón</b>							
desvinculación segura	n	s	s	s	s	s	s
bloqueo de cookies de encabezado	n	s	s	s	s	s	s
eliminación de direcciones/listas libres	n	n	n	s	s	s	s
cifrado de metadatos	n	n	n	s	s	s	s
finalizar cuando hay daños (apl. 32 bits)	n	n	n	OptIn	OptIn	OptIn	OptIn
finalizar cuando hay daños (apl. 64 bits)	n	n	n	OptOut	OptOut	OptOut	OptOut
<b>DEP</b>							
Soporte NX (i386)	n	OptIn	OptIn	OptIn	OptIn	OptIn	OptIn
Soporte NX (amd64, apl. 32 bits)	n	OptIn	OptIn	OptIn	OptIn	OptIn	OptIn
Soporte NX (amd64, apl. 64 bits)	n	AlwaysOn	AlwaysOn	AlwaysOn	AlwaysOn	AlwaysOn	AlwaysOn
<b>ASLR</b>							
<b>soporte de selección aleatoria</b>							
imágenes	n	n	n	OptIn	OptIn	OptIn	OptIn
pilas	n	n	n	OptIn	OptIn	OptIn	OptIn
montones	n	n	n	s	s	s	s
PEB/TEB	n	s	s	s	s	s	s
<b>entropía (bits)</b>							
imágenes	0	0	0	8	8	8	8
pilas	0	0	0	14	14	14	14
montones	0	0	0	5	5	5	5
PEB/TEB	0	4	4	4	4	4	4
<b>API</b>							
Soporte SetProcessDEPPolicy	n	n	s	n	s	s	s

Figura 6: Disponibilidad de mitigaciones de vulnerabilidades de seguridad en las SKU de servidor de Windows.

	Servidor 03 RTM	Servidor 03 SP1, SP2	Servidor 08 RTM	Servidor 08 R2 RTM
<b>SEH</b>				
SafeSEH	n	s	s	s
SEHOP	n	n	OptOut	OptOut
Soporte OptIn para SEHOP por proceso	n	n	n	s
<b>Montón</b>				
desvinculación segura	n	s	s	s
bloqueo de cookies de encabezado	n	s	s	s
eliminación de direcciones/listas libres	n	n	s	s
cifrado de metadatos	n	n	s	s
finalizar cuando hay daños (apl. 32 bits)	n	n	OptIn	OptIn
finalizar cuando hay daños (apl. 64 bits)	n	n	OptOut	OptOut
<b>DEP</b>				
Soporte NX (i386)	n	OptOut	OptOut	OptOut
Soporte NX (amd64, apl. 32 bits)	n	OptOut	OptOut	OptOut
Soporte NX (amd64, apl. 64 bits)	n	AlwaysOn	AlwaysOn	AlwaysOn
Soporte NX (ia64)	N/A	AlwaysOn	AlwaysOn	AlwaysOn
<b>ASLR</b>				
<i><b>soporte de selección aleatoria</b></i>				
imágenes	n	n	OptIn	OptIn
pilas	n	n	OptIn	OptIn
montones	n	n	s	s
PEB/TEB	n	s	s	s
<i><b>entropía (bits)</b></i>				
imágenes	0	0	8	8
pilas	0	0	14	14
montones	0	0	5	5
PEB/TEB	0	4	4	4
<b>API</b>				
Soporte SetProcessDEPPolicy	n	n	s	s

## Adopción de mitigación

La eficacia de las tecnologías de mitigación de vulnerabilidades de seguridad que se describen anteriormente depende en gran medida de su adopción por parte de las aplicaciones que se ejecutan en Windows. Esto significa que las aplicaciones deben tener incorporadas mitigaciones del compilador como GS habilitadas y también deben habilitar correctamente el soporte de mitigaciones de plataformas como DEP y ASLR, que actualmente requieren que las aplicaciones participen explícitamente. Si no se habilitan correctamente una o más de estas mitigaciones, esto hace que a los atacantes les resulte más fácil explotar las vulnerabilidades usando herramientas y técnicas de las que de otro modo no dispondrían.

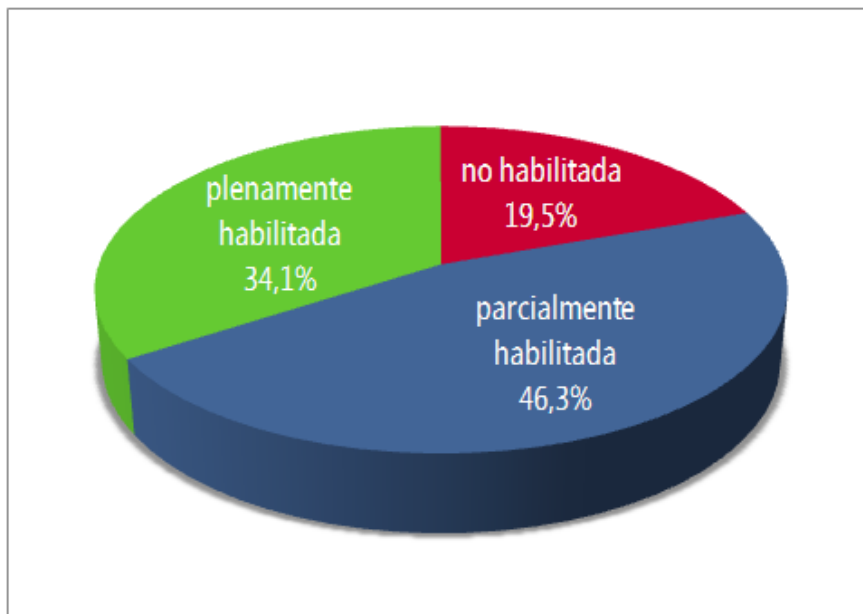
Estas implicaciones plantean una pregunta: si la eficacia depende de la adopción, ¿cuántas aplicaciones adoptan actualmente estas tecnologías de mitigación? Para entender mejor la respuesta a esta pregunta, realizamos una encuesta de las configuraciones de DEP y ASLR para las versiones más recientes de 41 aplicaciones para el consumidor populares que utilizan millones de usuarios en todo el mundo. Durante este proceso detectamos que el 71% de las aplicaciones encuestadas habilitaban plenamente el soporte de DEP pero sólo el 34% de las aplicaciones habilitaban plenamente el soporte de ASLR. En las siguientes secciones se proporciona un análisis detallado de estos datos.

## Adopción de ASLR

Antes de examinar los datos acerca de la adopción de ASLR, es importante entender de qué forma las aplicaciones realmente habilitan el soporte de ASLR. Para habilitar el soporte de ASLR, una aplicación debe vincular sus imágenes ejecutables (archivos EXE o DLL) con la marca `/DYNAMICBASE`. Esta marca le indica a las versiones aplicables del sistema operativo Windows que una imagen es compatible con ASLR. Windows no realiza una selección aleatoria de las imágenes a las que les falta esta marca cuando se ejecuta una aplicación y, por lo tanto, es posible que se carguen en la misma dirección. Esta falta de selección aleatoria reduce significativamente la eficacia de ASLR, porque le puede ofrecer a un atacante una posición predecible en el espacio de direcciones del programa, que puede usar para desarrollar una vulnerabilidad de seguridad. En ese sentido, *todas* las imágenes ejecutables que incluye una aplicación se deben vincular con esta marca para que ASLR sea lo más eficaz posible. Aunque la marca `/DYNAMICBASE` está habilitada de forma predeterminada en Visual Studio 2010, las versiones anteriores de Visual Studio requieren que los desarrolladores habiliten explícitamente esta marca en las configuraciones de sus proyectos. Estas opciones predeterminadas fueron diseñadas para garantizar la compatibilidad para las aplicaciones que hacen suposiciones acerca del diseño de espacio de direcciones de un programa.

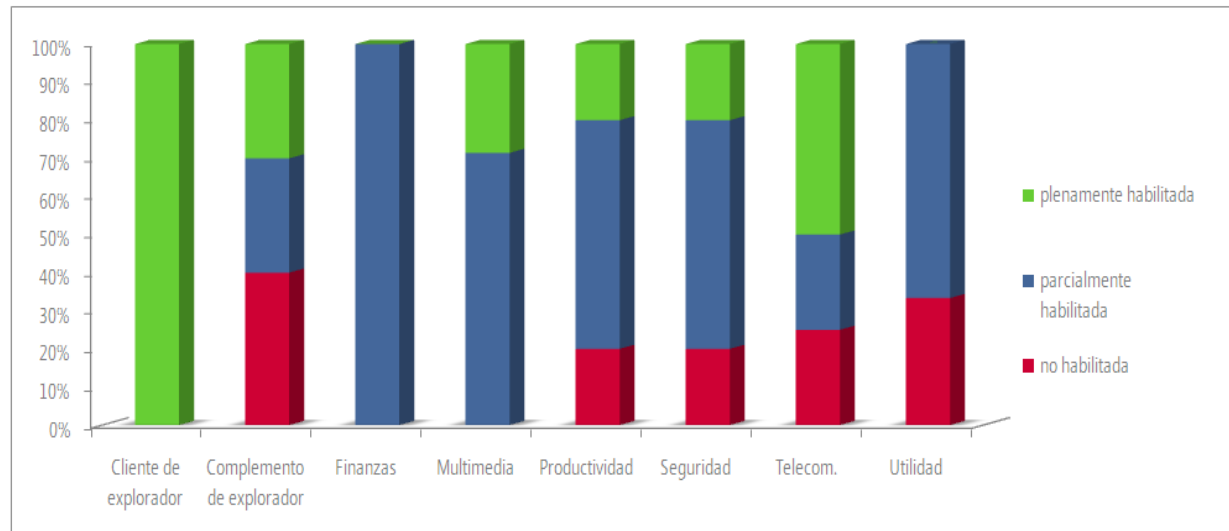
En la práctica, de las 41 aplicaciones encuestadas, el 34% habilitaban plenamente el soporte de ASLR, el 46% lo habilitaban parcialmente y el 20% no habilitaban el soporte para ninguna de las imágenes (Figura 7). Estos datos indican que muchas de las aplicaciones populares en este momento no habilitan plenamente el soporte de ASLR.

Figura 7: Porcentaje de aplicaciones que habilitan plenamente, habilitan parcialmente o no habilitan ASLR.



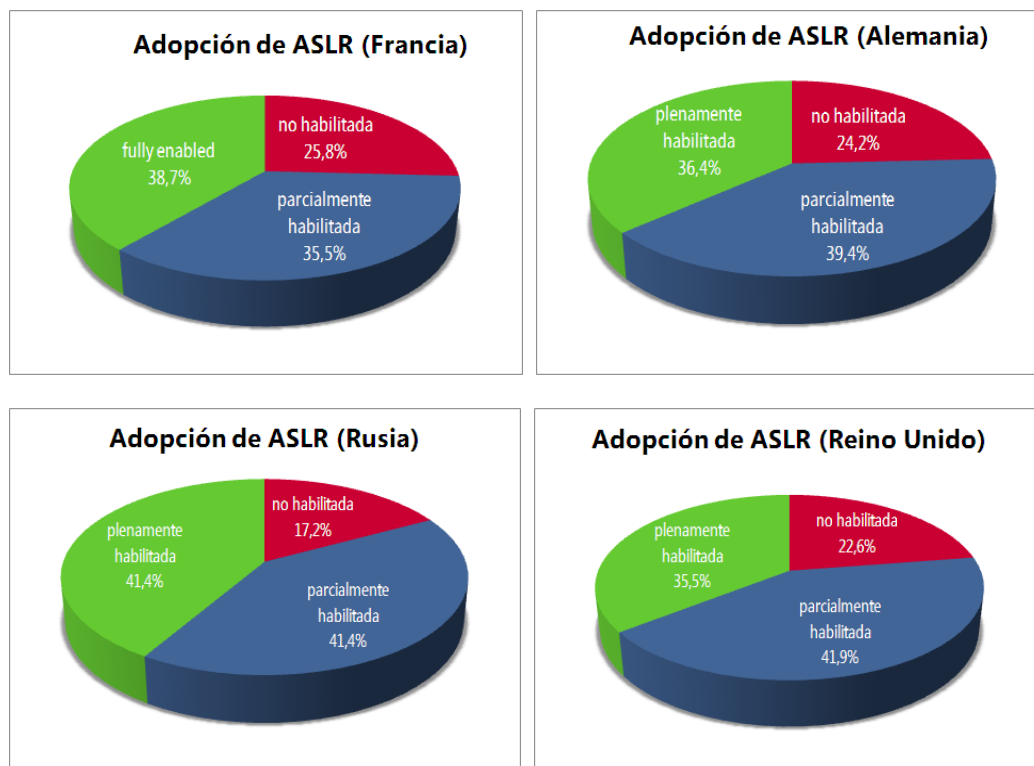
El diagrama de la Figura 8 ofrece una visión adicional acerca de la adopción de ASLR según los segmentos de mercado a los que pertenece cada aplicación. Se pueden hacer algunas observaciones dignas de mención a partir de estos datos. La primera observación es que todos los clientes de exploradores web (como Windows Internet Explorer®) a los que se les realizó la encuesta habilitan plenamente el soporte de ASLR. Desafortunadamente, el 70% de los complementos de los explorador encuestados no lo hacían, lo que significa que aunque ASLR debería ser eficaz en instalaciones de exploradores predeterminadas, la presencia de complementos del explorador probablemente reduzca la eficacia de ASLR. Una segunda observación es que sólo uno de los cinco productos de seguridad que se incluyen en este análisis habilitaban plenamente el soporte de ASLR. Esto es digno de mención, dado que los productos de seguridad están por naturaleza expuestos a datos que no son de confianza y la adopción limitada de ASLR podría, por lo tanto, hacer que a los atacantes les resulte más fácil explotar las vulnerabilidades en los productos de seguridad.

Figura 8: Porcentaje de aplicaciones que habilitan plenamente, parcialmente o no habilitan el soporte de ASLR por segmento de mercado.



También se analizó el tema de la relevancia geográfica de estos datos teniendo en cuenta el subconjunto de aplicaciones encuestadas cuya popularidad en Francia, Alemania, Rusia y el Reino Unido es conocida (un total de 31, 33, 29 y 31 aplicaciones, respectivamente). Estos resultados se muestran en la Figura 9.

Figura 9: Porcentaje de aplicaciones encuestadas que habilitan plenamente, parcialmente o no habilitan el soporte de ASLR (Francia, Alemania, Rusia y el Reino Unido).



Estos datos muestran claramente que la adopción de ASLR por parte de las aplicaciones en la mayoría de los segmentos de mercado ha sido muy lenta, a pesar de que la tecnología ha estado disponible desde hace más de cuatro años. Esto sugiere que muchas aplicaciones han tenido una postura de seguridad que es menos segura que lo necesario, lo que podría permitir que los atacantes exploten sus vulnerabilidades con mayor facilidad. Los clientes de exploradores web y, en menor grado, el software de telecomunicaciones (como clientes de mensajería instantánea) actualmente parecen ser las excepciones a esta regla. El hecho de que estas aplicaciones hayan adoptado ASLR más rápidamente que otras no debe resultar sorprendente, dada la cantidad de exposición directa que tienen estos tipos de productos a datos no confiables en Internet.



### Medidas a adoptar:

- Los desarrolladores de software deben comprobar que hayan configurado sus aplicaciones para que se compilen con la marca del vinculador /DYNAMICBASE.
- Los usuarios de software deben exigir que los proveedores de software compilen sus aplicaciones con la marca del vinculador /DYNAMICBASE.
- Comprobar la configuración de /DYNAMICBASE para las aplicaciones de interés por medio de la herramienta de Microsoft [SDL BinScope](#).

### Adopción de DEP

A diferencia de ASLR, DEP se habilita por proceso en lugar de por archivo de imagen. DEP se habilita automáticamente para todas las aplicaciones de 64 bits pero se debe habilitar explícitamente para las aplicaciones de 32 bits que se ejecutan en versiones de cliente del sistema operativo Windows. La razón por la cual las aplicaciones de 32 bits deben habilitar el soporte de DEP de forma explícita es garantizar la compatibilidad de las aplicaciones con el software que se escribió sin tener en cuenta la DEP. Hay varias maneras en que una aplicación puede elegir habilitar DEP, entre ellas la vinculación de archivos EXE con la marca /NXCOMPAT y llamando a la API SetProcessDEPPolicy en tiempo de ejecución. Una vez que activa DEP a través de cualquiera de estos mecanismos, queda habilitada de manera permanente. En otras palabras, una aplicación no puede deshabilitar la DEP más tarde mientras la aplicación se está ejecutando. Dado que DEP se habilita por proceso, no hay marcas especiales que se tengan que configurar para cada una de las imágenes ejecutables.

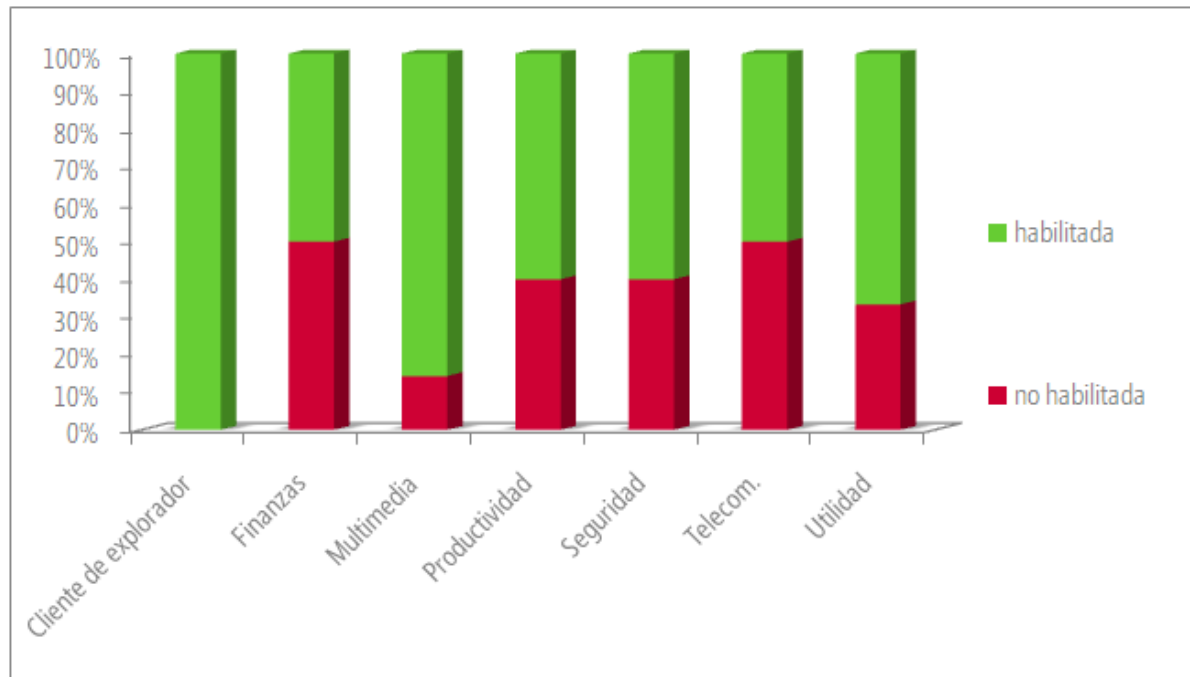
La simplicidad de la habilitación de DEP para una aplicación probablemente ha contribuido a la velocidad con la que la han adoptado los proveedores de software. El diagrama de la Figura 10 muestra que, de las aplicaciones encuestadas, el 71% habilitan DEP y el 29% no. Hay dos factores que probablemente hayan contribuido a que la mayoría de las aplicaciones encuestadas tengan DEP habilitada. El primer factor es que DEP ha estado disponible desde Windows XP Service Pack 2, mientras que ASLR sólo ha estado disponible desde Windows Vista®. Los desarrolladores de software han tenido más tiempo para adoptar DEP como tecnología y superar cualquier obstáculo al hacerlo. El segundo factor es que habilitar DEP para una aplicación es un proceso relativamente simple en la mayoría de los casos. Se debe tener en cuenta que los complementos de explorador se excluyen de estos datos porque sus configuraciones de DEP dependen de la configuración del explorador que los aloja.

Figura 10: Porcentaje de aplicaciones encuestadas que habilitan DEP.



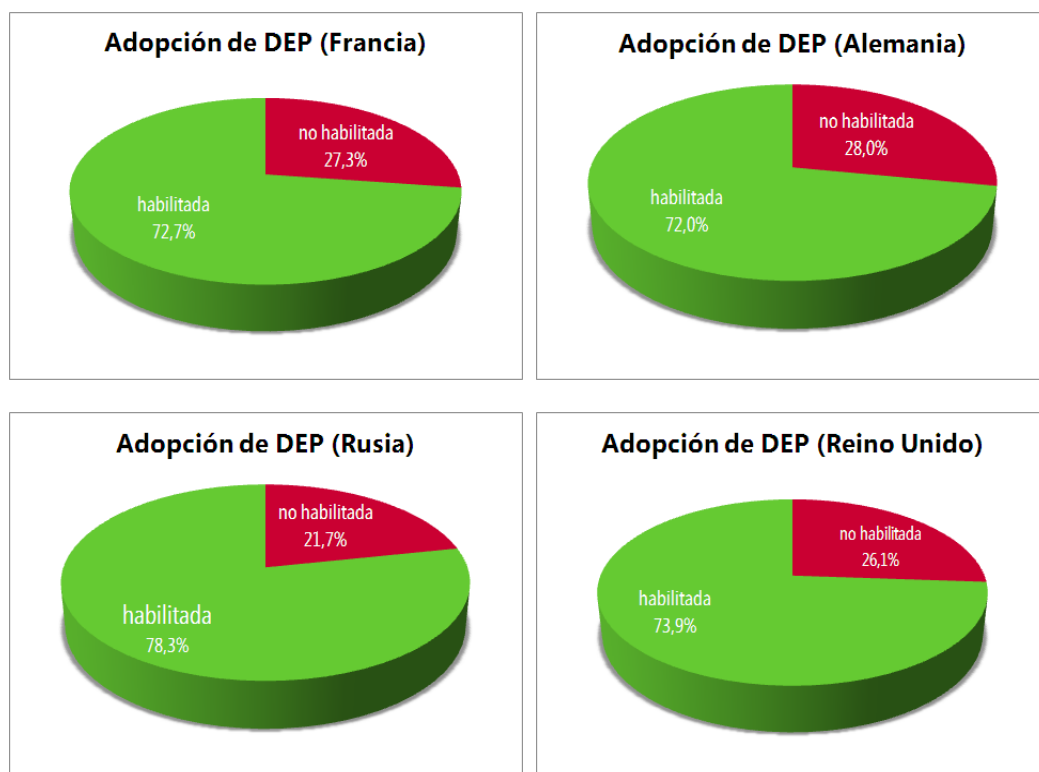
El diagrama de la Figura 11 ofrece un desglose de estos datos según el segmento de mercado al que pertenece cada aplicación. Al igual que con los datos sobre adopción de ASLR, los datos muestran que todos los clientes de exploradores encuestados habilitan plenamente el soporte de DEP. De forma similar, por lo menos la mitad de las aplicaciones de cada segmento de mercado habilitan el soporte de DEP.

Figura 11: Porcentaje de aplicaciones que habilitan o no habilitan DEP por segmento de mercado.



Se analizó el tema de la relevancia geográfica de estos datos teniendo en cuenta el subconjunto de aplicaciones cuya popularidad en Francia, Alemania, Rusia y el Reino Unido es conocida (un total de 22, 25, 23 y 23 aplicaciones, respectivamente, excluyendo los complementos de explorador). Estos resultados se muestran en la Figura 12.

Figura 12: Porcentaje de aplicaciones encuestadas que habilitan o no DEP en Francia, Alemania, Rusia y el Reino Unido.



#### Medidas a adoptar:

- Los desarrolladores de software deben comprobar que hayan configurado sus aplicaciones para habilitar DEP a través de SetProcessDEPPolicy o a través de la marca del vinculador /NXCOMPAT.
- Los usuarios de software deben exigir que los proveedores de software habiliten DEP para sus aplicaciones.

- Comprobar las configuraciones de DEP para las aplicaciones de interés utilizando el kit de herramientas de experiencia de mitigación mejorada (EMET), el explorador de procesos Sysinternals o el Administrador de tareas de Windows incorporado, como se muestra a continuación:

Figura 13: Administrador de tareas de Windows

Image Name	PID	CPU	Data Execution Prevention
cmd.exe *32	10748	00	Enabled

## Habilitación de mitigaciones en su software

Este documento muestra por qué las mitigaciones de vulnerabilidades de seguridad pueden ser una herramienta valiosa para ayudar a mitigar los riesgos planteados por vulnerabilidades conocidas y desconocidas. Microsoft cree firmemente en el valor de las tecnologías de mitigación de vulnerabilidades de seguridad y ha incorporado requisitos obligatorios en el proceso SDL para que los equipos de los productos usen estas funciones al desarrollar software. En la práctica, la eficacia de estas mitigaciones depende en gran medida de la adopción por parte de las aplicaciones que se ejecutan en el sistema operativo Windows. Las encuestas sobre aplicaciones populares para el consumidor indican que aunque muchas de las aplicaciones habilitan DEP, la mayoría no habilitan ASLR plenamente. Para mejorar esta situación, los proveedores de software necesitan realizar un esfuerzo coordinado para habilitar estas y otras tecnologías de mitigación en sus productos. Con este fin, Microsoft ofrece orientación diseñada para ayudar a los proveedores de software a habilitar diversas tecnologías de mitigación de vulnerabilidades de seguridad en sus productos:

<http://msdn.microsoft.com/en-us/library/bb430720.aspx>.

# Conclusión

---

Es imposible evitar por completo que se introduzcan vulnerabilidades durante el desarrollo de proyectos de software a gran escala. Los datos de la tendencia a largo plazo indican que se divulgan miles de vulnerabilidades de seguridad en toda la industria del software todos los años, la mayoría de las cuales son vulnerabilidades de alta gravedad en aplicaciones que son relativamente fáciles de explotar.

Microsoft ha avanzado en el uso del proceso SDL y la ciencia de la seguridad para reducir las vulnerabilidades e introducir mitigaciones de las amenazas contra el software y los servicios de Microsoft de formas que ayuden a proteger a los clientes de Microsoft y los usuarios de Internet. Los resultados de la investigación de este informe muestran que aunque muchas de las aplicaciones más populares del mundo aprovechan las mitigaciones de seguridad que se incorporan en los sistemas operativos Windows, sigue habiendo oportunidades para mejorar.

Los proveedores de software independientes y las organizaciones de desarrollo de software que actualmente no están usando el proceso SDL deben evaluar la posibilidad de aplicarlo para aprovechar sus beneficios.

Para obtener más información sobre SDL, la forma en que las organizaciones lo están utilizando y los beneficios del desarrollo seguro, visite:

<http://www.microsoft.com/sdl>

Para obtener más información sobre la ciencia de la seguridad en Microsoft visite:

<http://www.microsoft.com/msec>



One Microsoft Way  
Redmond, Washington  
98052-6399  
[microsoft.com/security](https://microsoft.com/security)