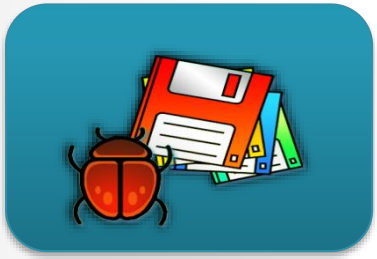
SDL

Ciclo de vida del desarrollo seguro

Evolución del cibercrimen

1986-1995



- LANs
- Primer virus PC
- Motivación: causar daño

1995-2003

- Era de Internet
- "Grandes Gusanos"
- Motivación: causar daño

2004+

- Ataques a SO, BD
- Spyware, Spam
- Motivación: económica

2006+

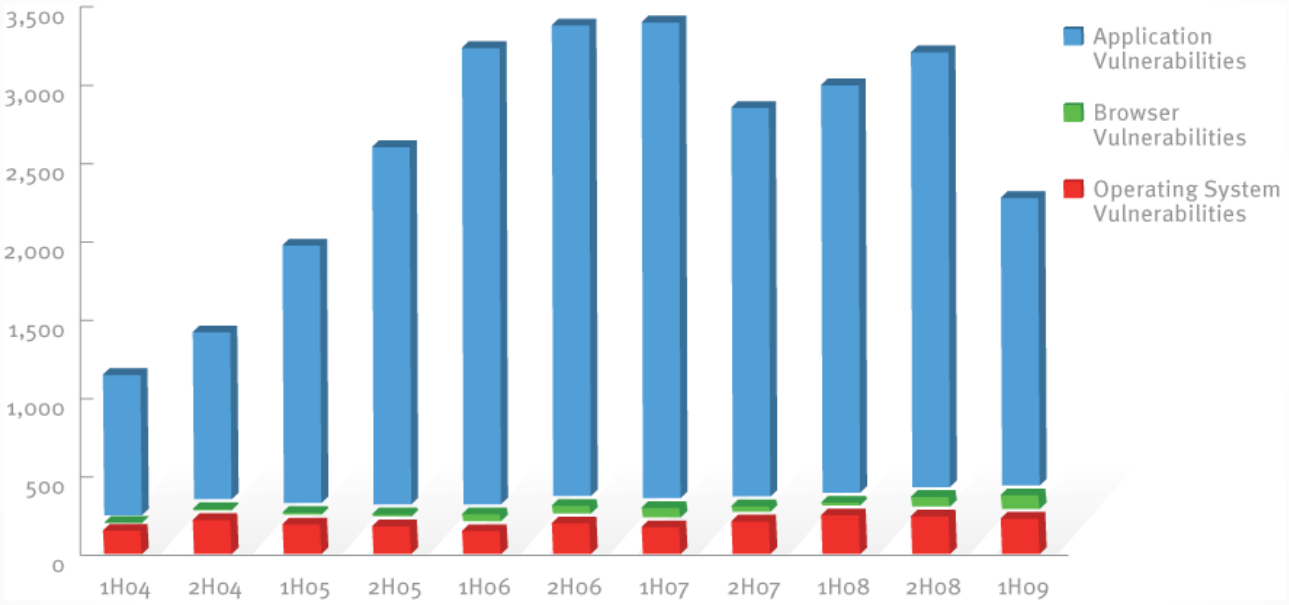
- Ataques dirigidos
- Ingeniería social
- Económica+política

→ ***Coste del cibercrimen en USA: Sobre $70B***

Precios de mercado 2007:	
Num. Tarjeta Crédito	$0.50 - $20
Identidad Completa	$1 - $15
Cuenta Bancaria	$10 - $1000

Ataques dirigidos a aplicaciones

Porcentaje de vulnerabilidades:
Sistema operativo vs Navegador vs Aplicación

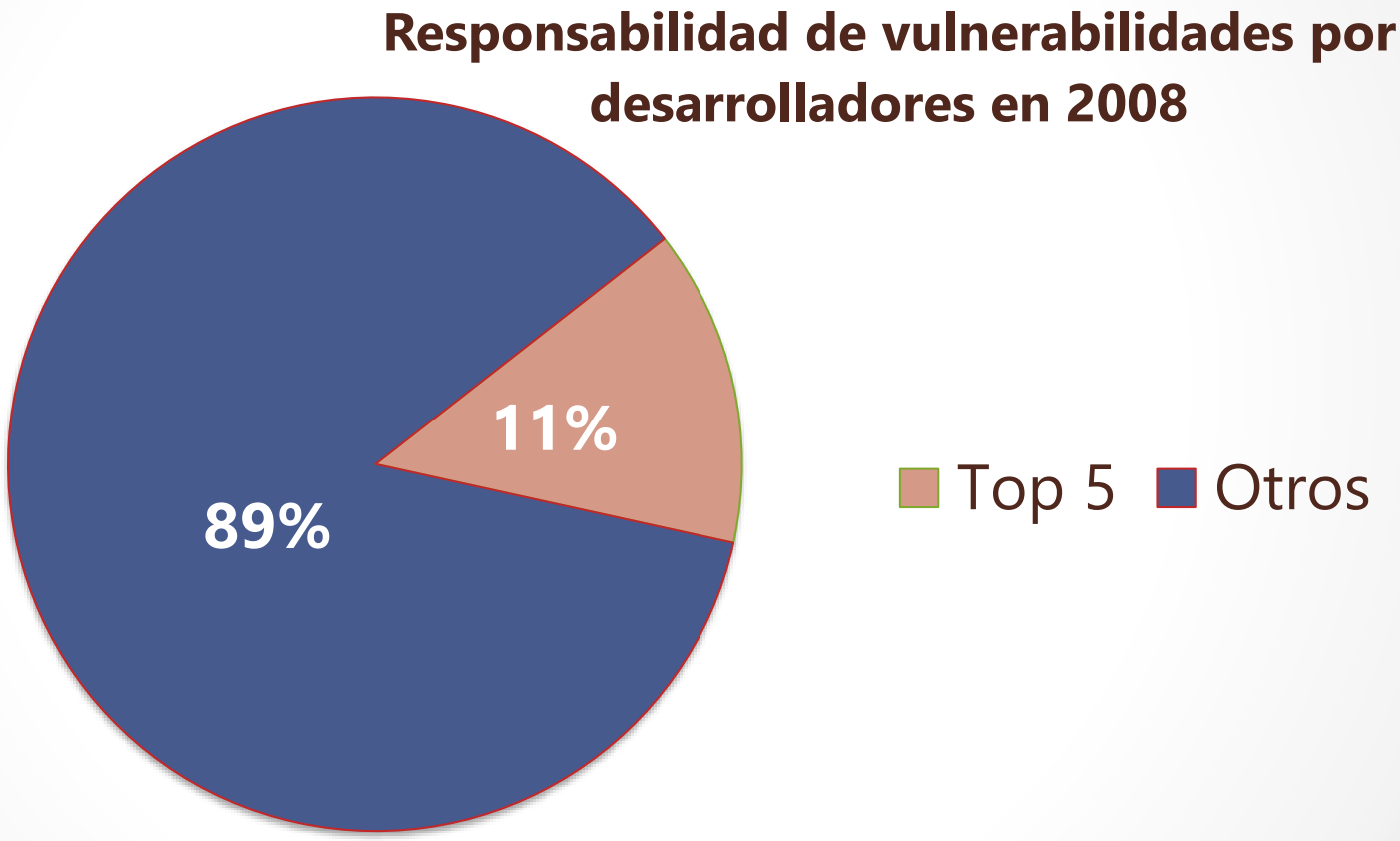


**Microsoft Security Intelligence Report V7*

90% de vulnerabilidades son explotables de forma remota

**IBM X-Force, 2008*

Problemas con desarrolladores pequeños



IBM X-Force 2008 Security Report

Evolución de la seguridad en MS

2002-2003

- Bill Gates escribe el memo "Trustworthy Computing" principios 2002
- "Windows security push" para Windows Server 2003
- Security push y RFS se extienden a otros productos

2004

- El equipo directivo de seguridad en Microsoft acuerda exigir SDL para todos los productos que:
  - Estén expuestos a un riesgo significativo y/o
  - Procesen datos sensibles

2005-2007

- SDL es mejorado
  - "Fuzz" testing
  - Análisis de código
  - Requisitos en el diseño de criptografía
  - Privacidad
  - APIs prohibidas
  - Etc.
- Windows Vista es el primer SO en pasar por el ciclo SDL completo

Ahora

- Optimización mediante retroalimentación, análisis y automatización
- Evangelización de SDL a la comunidad de desarrollo:
  - Ayuda en el proceso SDL
  - Modelo de optimización SDL
  - Red profesional SDL
  - Herramienta de modelado de amenazas SDL
  - Plantillas de proceso SDL

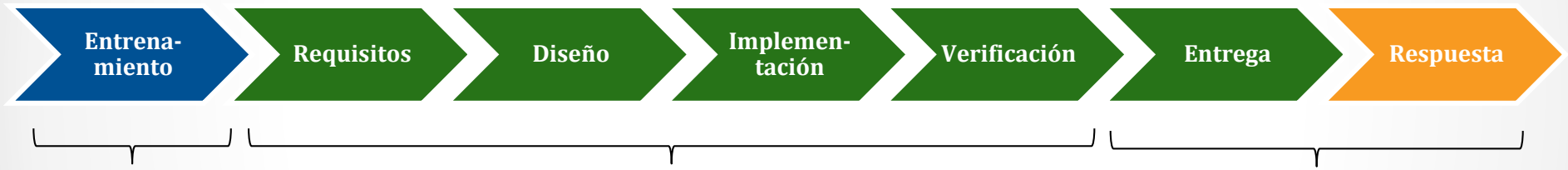
¿Qué aplicaciones deben seguir SDL?

- Cualquier producto utilizado habitualmente o implementado en empresas u organizaciones
- Cualquier producto que almacene o comunique datos sensibles o personales (identificativos)
- Cualquier producto en contacto con Internet u otras redes
- Cualquier producto que acepte y/o procese datos provenientes de una fuente no autentificada
- Cualquier funcionalidad que interprete tipos de fichero no protegidos (no limitados a administradores)
- Cualquier producto que contenga controles ActiveX y/o COM
- Todos los servicios online de Microsoft, MSN y Live.com que son accedidos por clientes externos

El SDL

Entregar software seguro requiere:

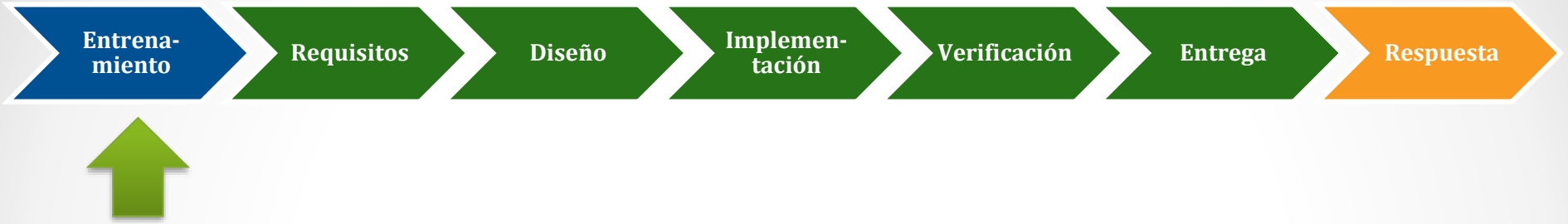
Compromiso directiva → SDL es política obligatoria en Microsoft desde 2004



Educación Tecnología y Proceso Responsabilidad

Mejoras continuas del proceso → ciclo de 6 meses

Requisitos Pre-SDL: Entrenamiento



Evaluar el conocimiento de la organización respecto a seguridad y privacidad; establecer el programa de entrenamiento de la manera necesaria

- Características
  - Contenido cubriendo diseño seguro, desarrollo, test y privacidad
- Frecuencia
  - Los empleados deben asistir a un número mínimo de clases al año
- Objetivos
  - Hitos respecto al entrenamiento (por ej. 80% del personal técnico entrenado antes de la entrega del producto)

Fase 1: Requisitos

Oportunidad para considerar la seguridad desde el inicio

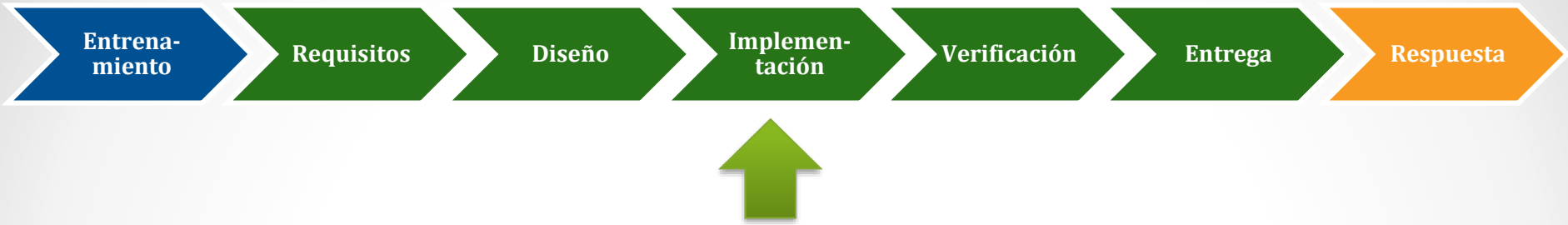
- El equipo de desarrollo identifica requisitos de seguridad y privacidad
- El equipo de desarrollo identifica a los encargados de seguridad y privacidad
- Se nombra un Asesor de Seguridad
- El Asesor de Seguridad revisa el plan del producto, hace recomendaciones y establece requisitos adicionales si es necesario
- Forzar el uso de un sistema de seguimiento de bugs y asignación de trabajos
- Definir y documentar objetivos de seguridad y privacidad

Fase 2: Diseño

Definir y documentar la arquitectura y componentes críticos de seguridad

- Identificar técnicas de diseño (layering, código gestionado, minimización del privilegio y superficie de ataque)
- Documentar la superficie de ataque y limitar en los valores por defecto
- Definir características de seguridad adicionales particulares al producto
  - Tests de cross-site scripting (XSS)
  - Deshabilitar la criptografía débil
- Modelado de amenazas
  - Revisión sistemática de las características y arquitectura respecto a la seguridad
  - Identificar amenazas y mecanismos de minimización de daño
- Requisitos específicos para servicios online

Fase 3: Implementación



Revisión completa – para determinar procesos, documentación y herramientas necesarias para garantizar una instalación y operación seguras

- Especificación de herramientas y opciones de compilación aprobadas
- Análisis estático (PREFix, /analyze (PREfast), FXCop)
- APIs prohibidas
- Uso de protecciones del SO "en profundidad" (NX, ASLR y HeapTermination)
- Requisitos específicos de servicios online (XSS, inyección SQL, etc.)
- Considerar otras recomendaciones (por ej. Standard Annotation Language (SAL))

Fase 4: Verificación

Inicio tan pronto como sea posible, al tener el código completo

- Iniciar la planificación de respuesta de seguridad – incluir planes de respuesta para informes de vulnerabilidad
- Reevaluar la superficie de ataque
- Fuzz testing – ficheros, controles instalables y código de red
- Realizar un "security push" (según sea necesario, cada vez menos)
  - No es un sustituto del trabajo de seguridad realizado durante el desarrollo
  - Revisión de código
  - Pentesting y otras pruebas de seguridad
  - Revisar diseño y arquitectura frente a nuevas amenazas
- Requisitos específicos para servicios online

Fase 5: Entrega – Plan de respuesta

Definición clara de la política de soporte – de acuerdo con las políticas corporativas de MS

- Proporcionar un Plan de Respuesta frente a un Incidente de Seguridad en el Software (SSIRP)
  - Identificar responsables y recursos para responder a eventos
  - Información de contacto 24x7x365 para 3-5 ingeniería, 3-5 marketing, y 1-2 gestión
- Asegurarse de la capacidad de mantener todo el código incluyendo el licenciado de terceras partes.

Fase 5: Entrega – Revisión Final de Seguridad

Verificar el cumplimiento de los requisitos SDL y que no hay vulnerabilidades de seguridad conocidas

- Proporciona una perspectiva independiente acerca del estado del producto
- La RFS no es:
  - Un pentest – no se permite “probar y parchear”
  - La primera vez que la seguridad se revisa
  - Una autorización
  - Concepto clave: esta fase sirve como factor para determinar si la aplicación debe ser entregada o no, evitando ser un cajón de sastre para todo el trabajo que no se haya realizado previamente

Fase 5: Entrega – Archivado

El plan de respuesta de seguridad está completado

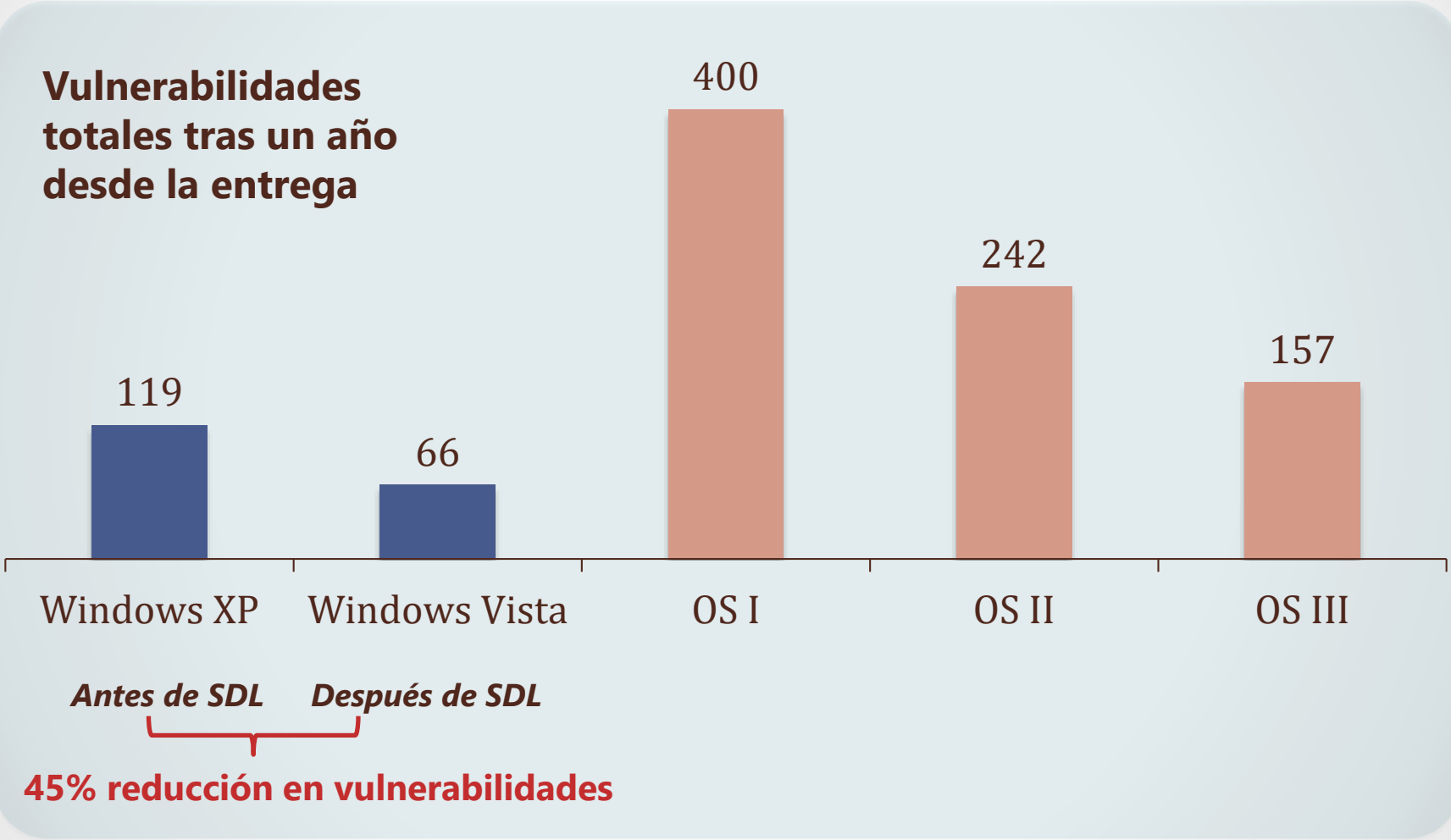
- La documentación del cliente está actualizada
- Archivar el código, símbolos y modelos de amenaza de la versión RTM en un repositorio central
- Completar las autorizaciones finales en Checkpoint Express – validar el cumplimiento de las políticas de seguridad, privacidad y corporativas

Requisito Post-SDL: Respuesta

"Planifica el trabajo, trabaja el plan…"

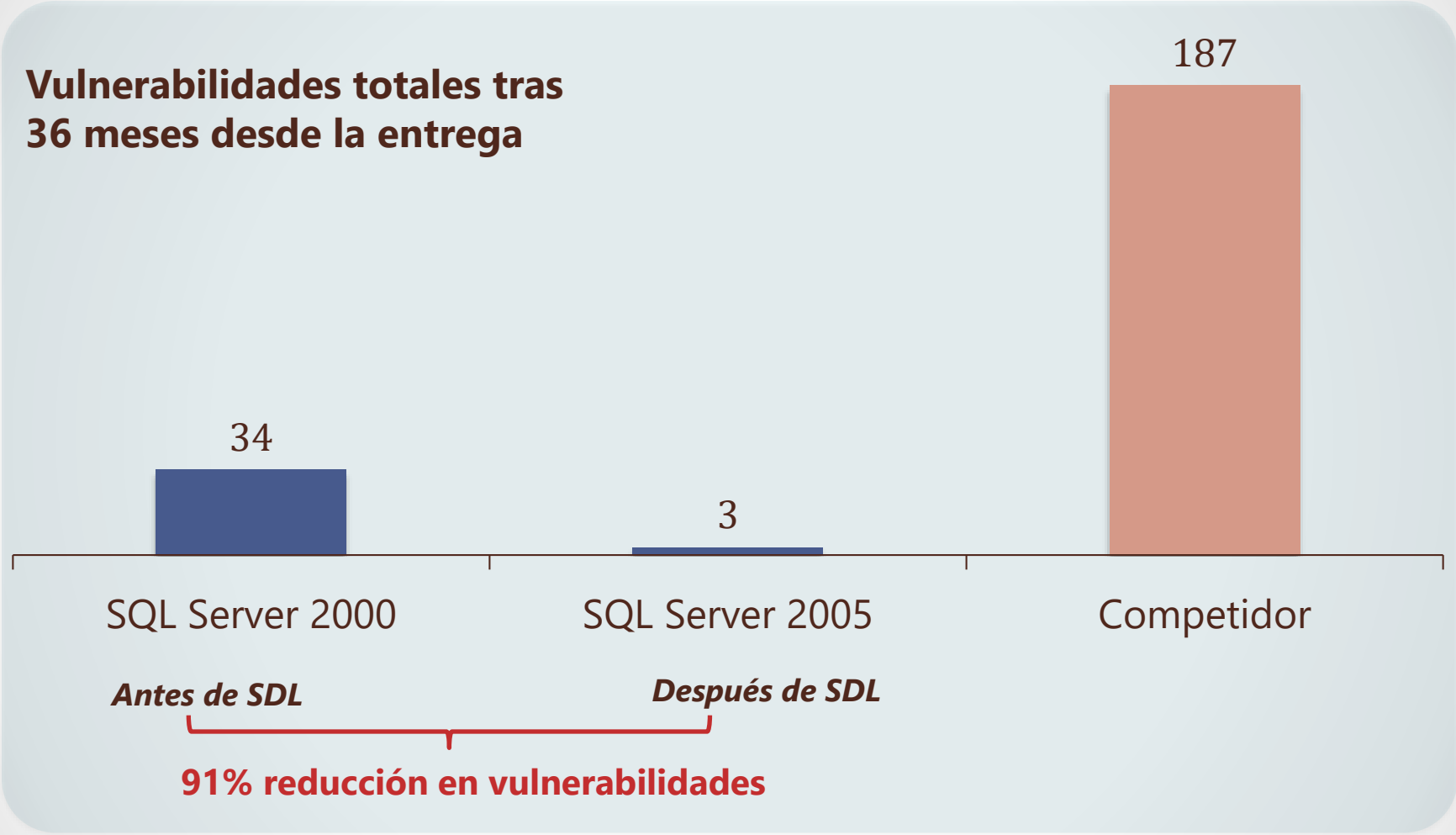
- Ejecución de las tareas de respuesta establecidas durante las fases de planificación de respuesta de seguridad y entrega

Microsoft SDL y Windows



**Windows Vista One Year Vulnerability Report, Microsoft Security Blog 23 Jan 2008*

Microsoft SDL y SQL Server



*Analysis by Jeff Jones (Microsoft technet security blog)

Conclusiones

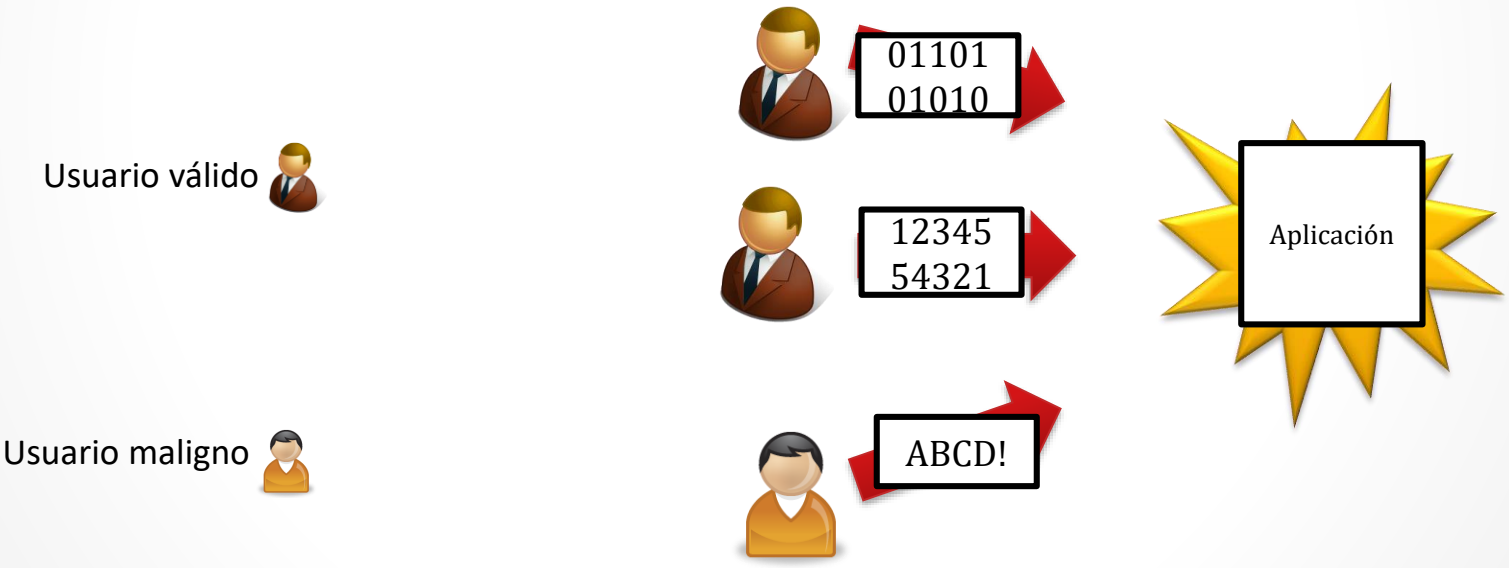
- Los ataques van dirigidos a las aplicaciones
- El SDL consiste en incorporar seguridad en el desarrollo de software y en la cultura corporativa
- Se han obtenido resultados medibles en el caso del software de Microsoft
- Microsoft se ha comprometido a que el SDL esté disponible y sea accesible

Fuzz Testing

Introducción a Fuzz testing

- Consiste en introducir datos malformados y/o erróneos en la aplicación comprobando la reacción de la misma

- Si la aplicación falla es que se ha encontrado una vulnerabilidad potencial



Fuzz testing

Ventajas

- Las vulnerabilidades identificadas suelen ser de naturaleza severa
- Se puede automatizar con facilidad

Desventajas

- No puede identificar vulnerabilidades que no causan una excepción:
  - filtración de información
  - fallo en la criptografía
  - etc.

No debe utilizarse como remplazo de otras técnicas de evaluación y no debe ser considerada una panacea, es un complemento

Tipos de fuzz testing

Aleatorio

- Se introducen datos en la aplicación que cambian de forma puramente aleatoria

Kevin, 123, Microsoft, Internet, SQLServer, k3v1n11, AAAAAAAAAAA, 00000_, !@#%^

Inteligente

- Se modifican valores específicos en función de la experiencia previa y/o el comportamiento esperado

(AAA) 123-4567, (123) 123*1234, (000 123-4567, (<empty>)123-4567, (123) 456---5678, ((123)4567890,(AAAAAAAA) 123-4567, (<U+07C0>23) 123-4579

Realización de Fuzz tests

1. Determinar los puntos de entrada a la aplicación

2. Ordenar dichos puntos de entrada en función del privilegio y la accesibilidad

3. Crear e introducir datos malformados en la aplicación para evaluar los puntos de entrada que están en riesgo
   - Intentar automatizar el proceso en la medida de lo posible

4. Analizar cualquier fallo, excepción o error inesperado o no gestionado debidamente para identificar vulnerabilidades severas

5. Reparar y volver a comprobar

Ampliación

Microsoft tiene una gran cantidad de información disponible sobre el SDL (en inglés)

- Este libro detalla el SDL en profundidad (en inglés).