Seguridad en el Desarrollo de Aplicaciones

Nuestro curso presenta un **abordaje inicial** al tema de la **seguridad de aplicaciones de software** desde una perspectiva teórico-práctica, discutiendo los tipos tipos de fallos (vulnerabilidades) de seguridad que se pueden observar en las apliaciones de software.

Se discuten desafíos para varios tipos de vulnerabilidades de software y prácticas seguras de desarrollo de software. Utilizamos herramientas de análisis estático y dinámico para la detección y corrección de dichas vulnerabilidades.

Los ejemplos, problemas, o fallas (vulnerabilidades) se presentan en syntaxis del **lenguaje C** (*típicamente conocido por ser suceptible a fallas de seguridad y a ataques*), y ocacionalmente en otros lenguajes como **Python**, **Shell**, **Java**, **OCaml**, etc.

¿Lo que vamos a Aprender?

- Principios de diseño seguro (**Saltzer and Schroeder**):
    - economy of mechanism
    - fail-safe defaults
    - complete mediation
    - open design
    - separation of privilege
    - least privilege
    - least common mechanism
- Cómo los principios de diseño seguro se evidencian en el desarrollo de aplicaciones de software?
- Técnicas de validación de datos de entrada (**input validation**)
- Cómo las vulnerabilidades son registradas (**CVE, CWE**)
- Tipos de ataques informáticos:
    - Cross domain attacks
    - command injection
    - SQL injection
    - Clickjacking
- Memory vulnerabilities:
    - buffer overflow
    - heap overflow
    - Integer overflow
    - pointer overwrites
- Técnicas de análisis estático y dinámico para prevención de problemas de seguridad.

    - model-checking
    - symbolic execution
    - concolic execution
    - taint analysis

- Introducción a técnicas criptográficas.

Metodología

Parte sincrónica

- Slides
- Discusión en clase
- Demostración de algún concepto, error o ataque

Parte asincrónica

- Vídeos sobre material complementario ó técnica
- Demostración de programas ó aplicaciones software
- Tutorial de aprendizaje
- Talleres
    - Lectura de un artículo (preguntas y respuestas)
    - Dado un programa en C, identificar tipos de errores
    - CVE, CWE

Evaluación

# taller	%	Tema	Fecha-Publicación	Fecha Entrega	Hora
1	10%	Seguridad de Chromium	22-04-2025	29-04-2025	6PM
2	10%	Validación de datos de entrada	29-04-2025	06-05-2025	6PM
3	10%	Analizar un CVE de prioridad alta	06-05-2025	13-05-2025	6PM
4	10%	Lectura: Make Least Privilege a Right	13-05-2025	20-05-2025	6PM
5	10%	Presentación tarea 3	20-05-2025	27-05-2025	6PM
6	10%	Práctico: fallas de seguridad en C/C++	27-05-2025	03-06-2025	6PM
7	10%	Práctico: fallas de seguridad en C/C++	03-06-2025	10-06-2025	6PM
8	10%	Memory corruption	10-06-2025	17-06-2025	6PM
9	20%	Criptografía	17-06-2025	23-06-2025	6PM

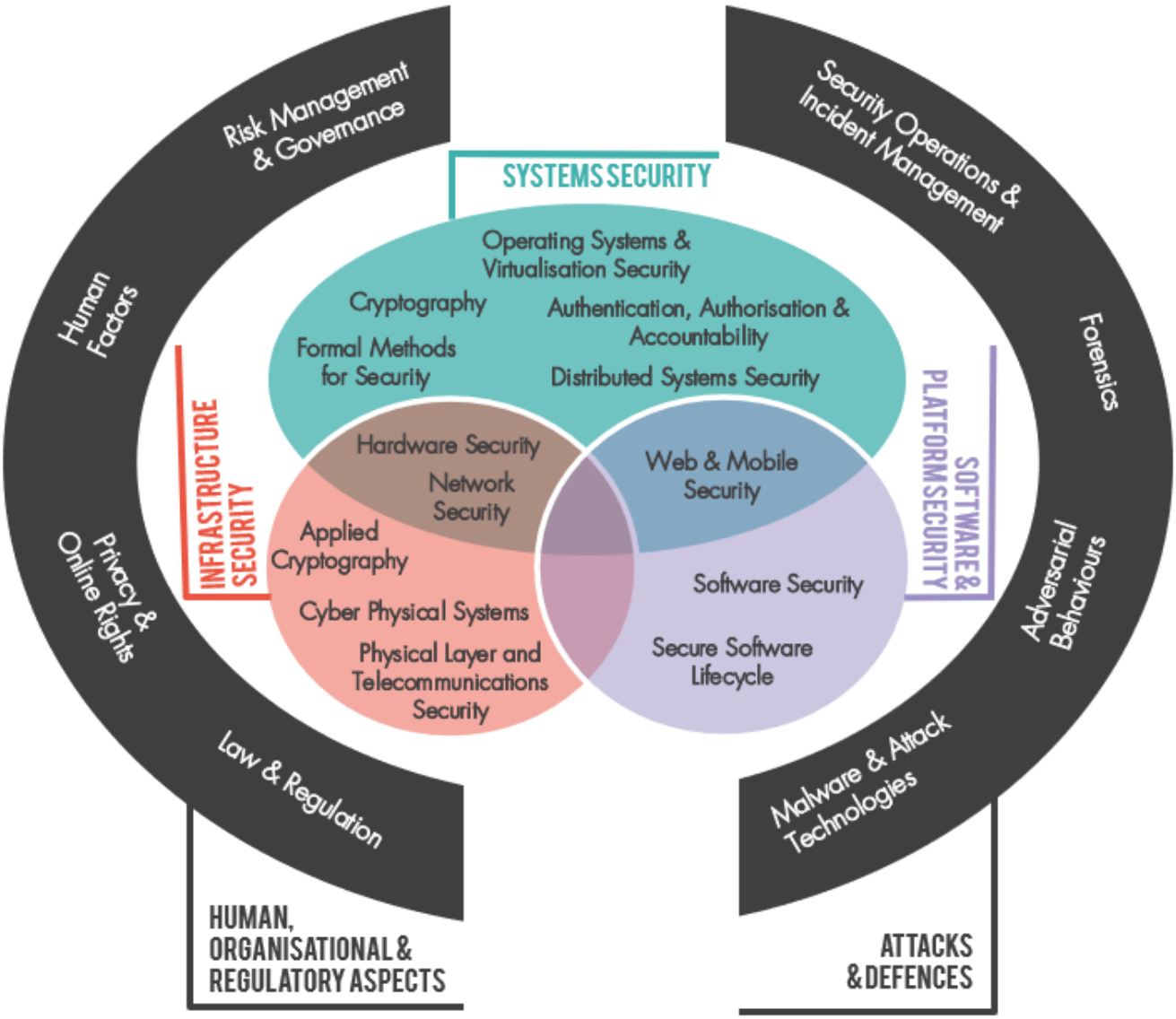
Software Security Landscape

Panorama de la Seguridad del Software

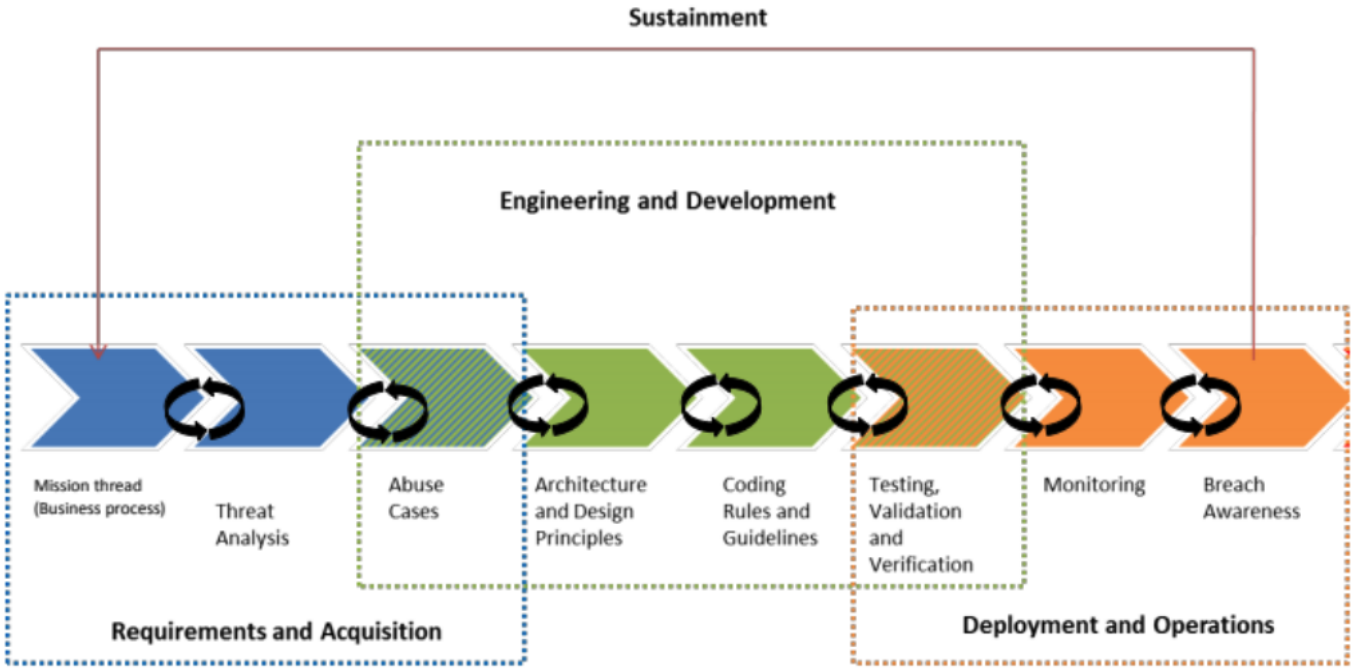
- problemas de seguridad del software
- costo de las vulnerabilidades de seguridad
- variedad de fallos de seguridad
- causas del software inseguro
- principios de diseño seguro
- clasificación y registro de vulnerabilidades

Security Body of Knowledge

Source: cybok.org



Security in Software Lifecycle



Our focus: Design and development Source: Mark Sherman, CERT / SEI Webinar

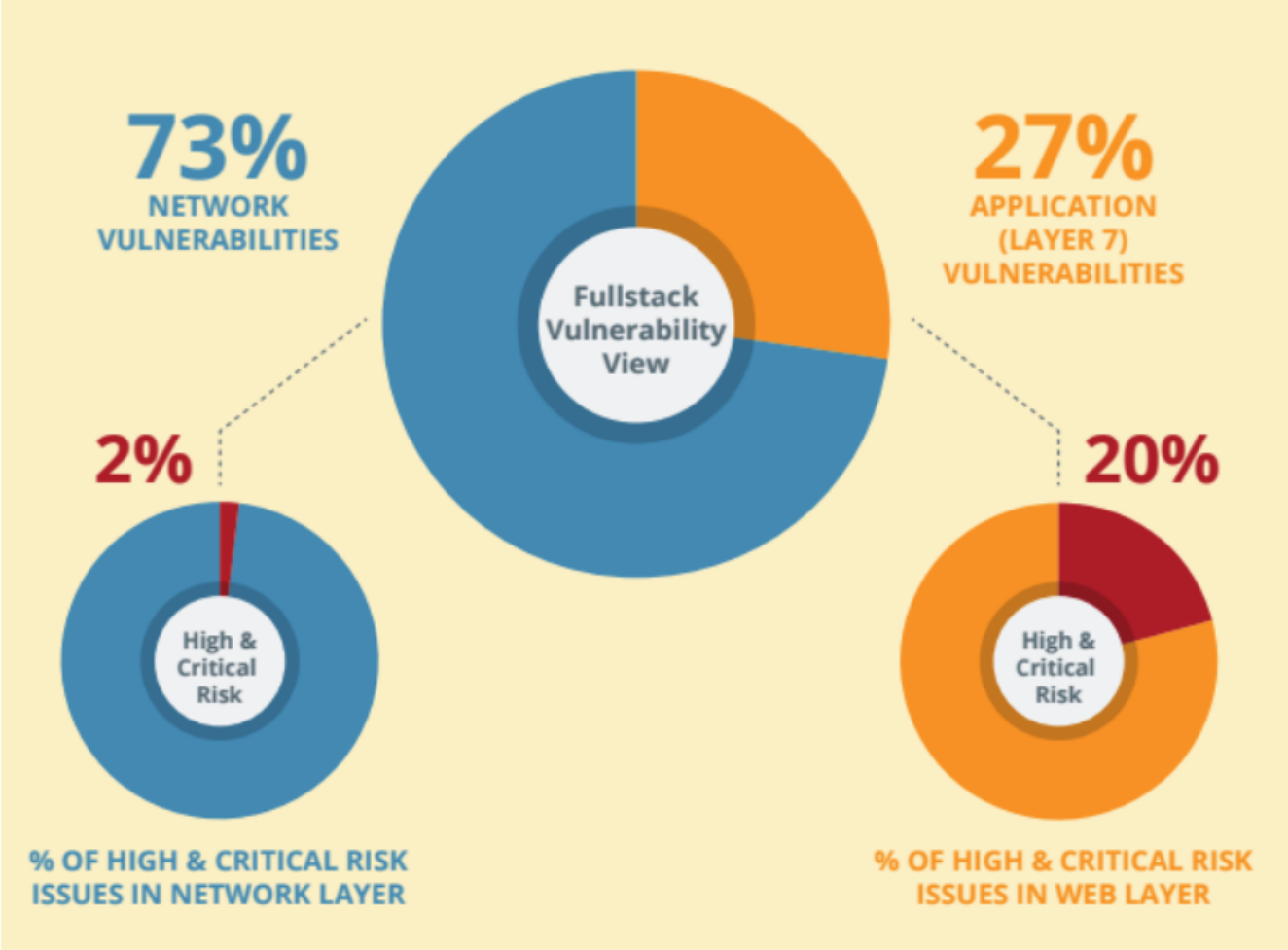
Problemas de Seguridad del Software

- Comprender las vulnerabilidades y sus causas
- Prevenirlas escribiendo software seguro
- **Aprender**:
  - Principios sólidos
  - Técnicas específicas y patrones de codificación
- Herramientas para detección, análisis y pruebas
- **Objetivo**: Ingeniería de software orientada a la seguridad

---

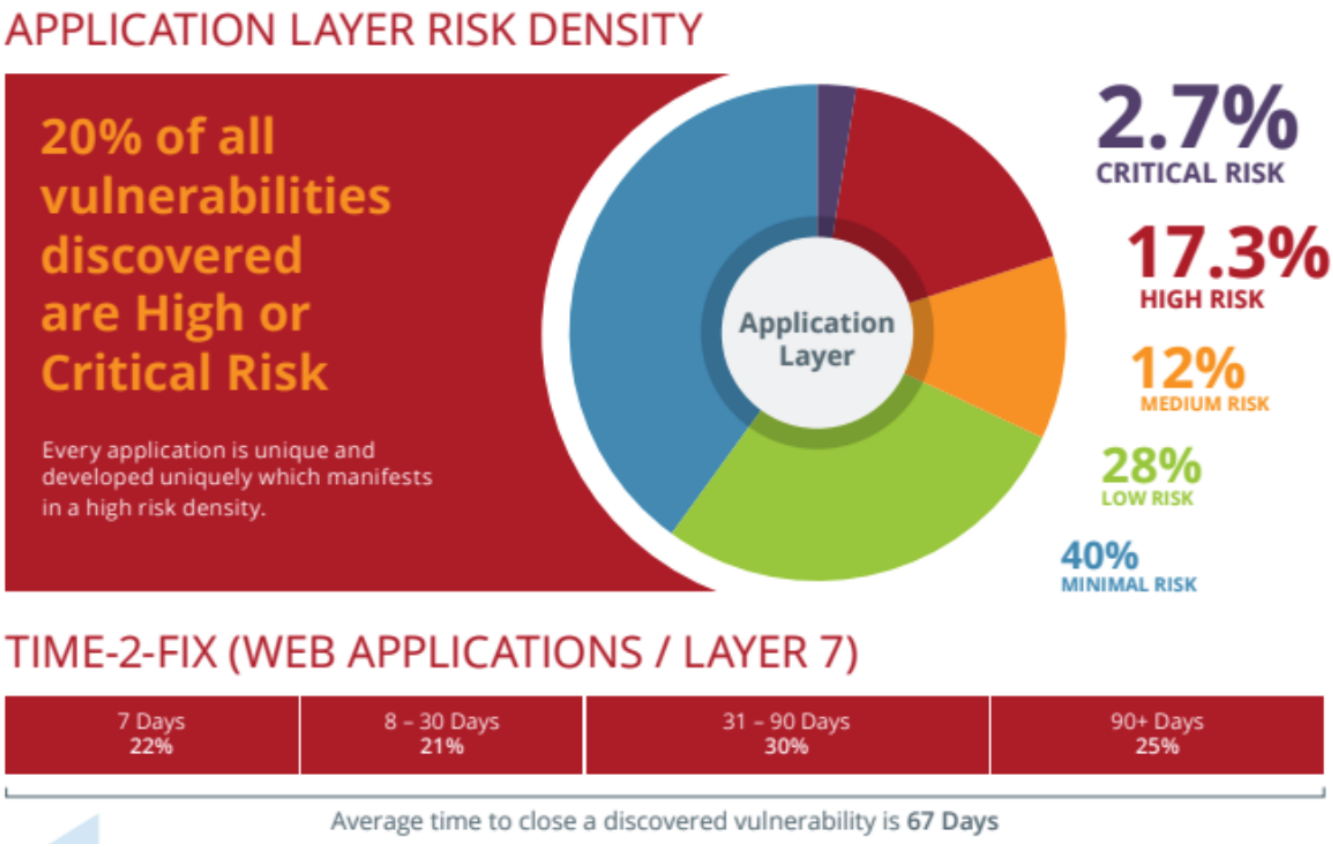
Location of Vulnerabilities

Web Applications vs. Network Layer



Source: Edgescan 2018 Vulnerability Statistics

Vulnerabilities by Risk



Vulnerabilities by Type (Web)

Costo de las Vulnerabilidades de Seguridad

¿Cuánto Tiempo Toma Corregirlas?

Estadísticas: Dan Cornell, RSAConf. 2012

- **9.6 minutos** para XSS almacenado
- **16.2 minutos** para XSS reflejado
- **84 minutos**: XSS almacenado/reflejado (total)

- **No importa**: La visión general
- **Todas las anteriores**

Remediación: ¡No es Solo Corregir un Bug!

- Se necesita:
  - Configurar el entorno de desarrollo
  - Corregir la vulnerabilidad
  - Confirmar la corrección
  - Realizar pruebas funcionales
  - Desplegar
  - **+ Sobrecarga**
- El costo total es mucho más que solo la corrección en sí

Costo del Cibercrimen

- **Informe de McAfee**: 600 mil millones de dólares (2018)
- **Informe de Accenture, 2017 (250 empresas):**
  - 11.7 millones de dólares por empresa
  - ¡130 brechas exitosas por empresa!
  - Ambas cifras aumentaron aproximadamente **25%** en un año
- **50 días en promedio**: Resolver un ataque interno
- **23 días en promedio**: Resolver un ataque de ransomware

Costos Directos de los Incidentes

	SMB		Enterprise	
	Proportion of business incurring this expense	Typical losses	Proportion of business incurring this expense	Typical losses
Professional services	88%	$11K	88%	$84K
Lost business opportunities	32%	$16k	29%	$203K
Down-time	34%	$66K	30%	$1.4M
Total expected typical damage	$38K		$551K	

Encuesta a **5500 empresas, 2015**
Fuente: Kaspersky Lab: Informe Especial sobre Riesgos de Seguridad Informática, 2015

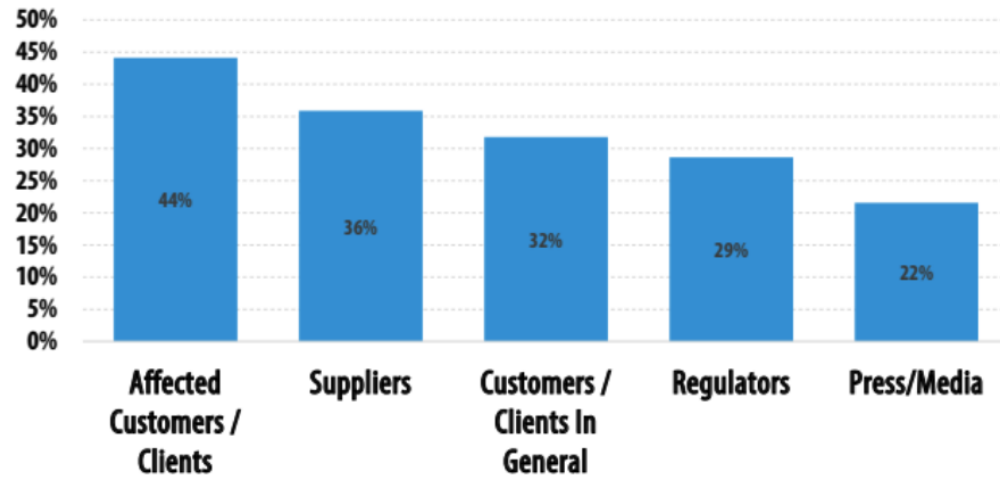
Costos Indirectos de los Incidentes

	SMB		Enterprise	
	Proportion of business incurring this expense	Typical losses	Proportion of business incurring this expense	Typical losses
Staffing	41%	$5.5K	40%	$52K
Training	47%	$5k	53%	$33K
Systems	54%	$7K	54%	$75K
Total expected indirect spend	$8K		$69K	

Gastos para prevenir incidentes en el futuro
Fuente: Kaspersky Lab: Informe Especial sobre Riesgos de Seguridad Informática, 2015

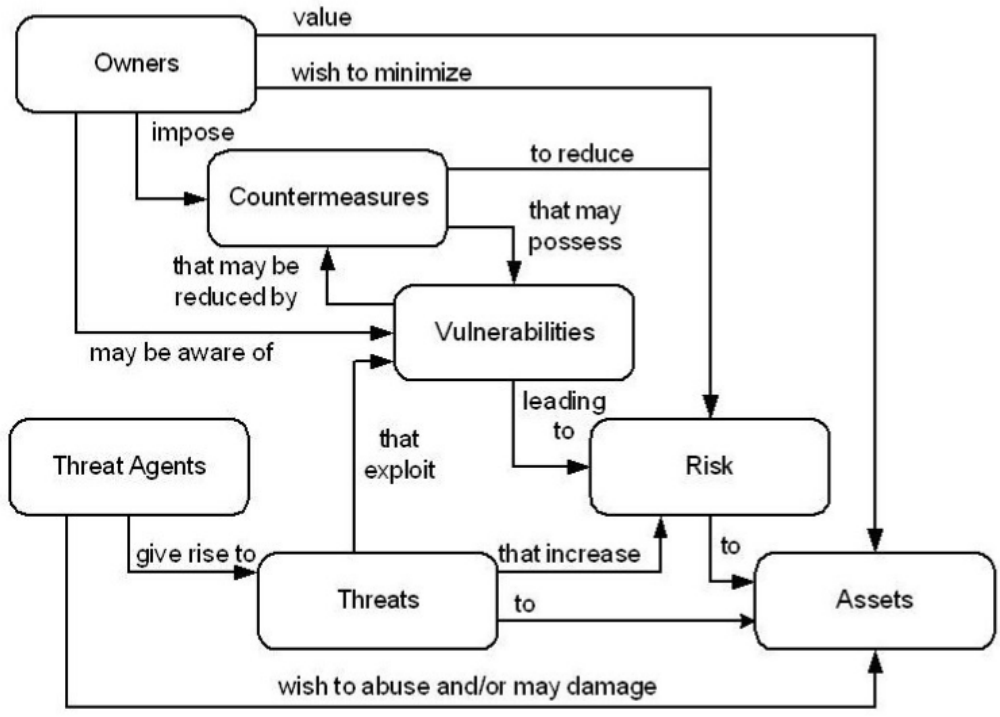
Divulgación y Reputación

- Es necesario divulgar las brechas a:
  - Partes interesadas
  - Autoridades
  - Público



Gama de Fallos de Seguridad

Vulnerabilidades en la Imagen



De la norma **ISO 15408**

¿Qué es una Vulnerabilidad?

- Una debilidad que puede ser explotada por un atacante para realizar acciones no autorizadas
- **No autorizado** = Violación de la política de seguridad
- Definición muy amplia
- En software, nos enfocamos en algunas clases comunes de vulnerabilidades
  - 5 vulnerabilidades principales
  - Ver **Área de Conocimiento de Seguridad de Software (Cyber Security Body of Knowledge)**

(i.) Vulnerabilidades de Corrupción de Memoria

- También llamadas **vulnerabilidades de gestión de memoria**
- Se encuentran en lenguajes imperativos donde la **asignación/liberación** de memoria es responsabilidad del programador
  - **Vulnerabilidad espacial** (acceso fuera de límites)
  - **Vulnerabilidad temporal** (uso de memoria que ya no está en uso)
- **Ataques solo de datos**
- **Corrupción de código (inyección de código)**
- **Secuestro del flujo de control**: Sin necesidad de inyección
- **Fugas de información**

(ii.) Generación de Salida Estructurada (Inyección de Código)

- Construcción dinámica de salida en alguna sintaxis (**consulta SQL**, página **HTML**)
- Si se construye sin control a partir de entrada no confiable → los datos pueden interpretarse como

comandos (inyección SQL, inyección de comandos/scripts)
- Casos problemáticos:
  - Sublenguajes con sintaxis distinta (**JavaScript** incrustado en **HTML**)
  - Procesamiento en múltiples fases (inyección almacenada o de orden superior)

(iii.) Condición de Carrera (en concurrencia)

- La concurrencia puede llevar al no determinismo
- El atacante puede controlar el orden o el momento de ejecución de actores concurrentes, violando el objetivo de seguridad
- **Del momento de verificación al momento de uso**: invalida una condición que se asumía ya verificada (**mediación completa**)
- **Condición de carrera en el sistema de archivos**: acceso privilegiado por un usuario no autorizado
- **Corrupción del estado de sesión web** por hilos mal sincronizados

(iv.) Vulnerabilidades en APIs

- Interfaz de comunicación entre componentes (por ejemplo, programa y biblioteca)
- Toda API tiene **condiciones para su uso correcto**: un **contrato** que debe observarse
- **Violar el contrato** puede llevar al sistema a un estado de error explotable
- **Problema común**: mal uso de APIs criptográficas (la flexibilidad hace

difícil el uso correcto)

(v.) Vulnerabilidades de Canal Lateral

- Transmiten información sobre la ejecución del programa a través de **detalles de implementación por debajo del nivel de abstracción del software** (p. ej., energía, tiempo, estado de la microarquitectura)
- **Canal encubierto**: el atacante también controla el programa que se comunica por ese canal
- **Ataque usual**: fuga de información
- También puede resultar en **ataques por inyección de fallos** (forzando al hardware fuera de su rango normal)

Conclusión: Fallos como Violaciones de Contrato

- Las vulnerabilidades son fallos que pueden **llevar a la violación de políticas de seguridad** (es decir, un tipo particular de especificación/contrato)
- No hay necesariamente una relación uno a uno
- **Formalizar los objetivos de seguridad** en detalle puede ayudar tanto en la **prevención** como en la **detección** de vulnerabilidades