Módulo 1: Fundamentos de seguridad computacional

Lección 2: Anatomía de las Vulnerabilidades

Objetivos de la Lección

Al terminar esta sesión, el estudiante podrá:

1. **Identificar** vulnerabilidades en hardware, software, datos, lógica, procesos, protocolos y factores humanos.
2. **Distinguir** entre fallos lógicos (diseño/implementación) y fallos humanos (comportamiento/operación).
3. **Analizar** el impacto potencial de cada tipo de vulnerabilidad sobre la confidencialidad, integridad y disponibilidad (CIA).

Introducción a la Lección

Una vulnerabilidad es cualquier debilidad aprovechable que podría comprometer un activo digital. Comprender su anatomía permite anticipar amenazas y priorizar controles. En esta lección clasificaremos las vulnerabilidades por su origen técnico y humano, y evaluaremos su impacto mediante la triada CIA y el marco CVSS (Common Vulnerability Scoring System).

Desarrollo del Tema

¿Qué es una vulnerabilidad y como trabaja la gestión de estas?

- **Enlace Video: https://youtu.be/VkS56Zz-iEo?si=Fj8F84AXcMKCbxxH**

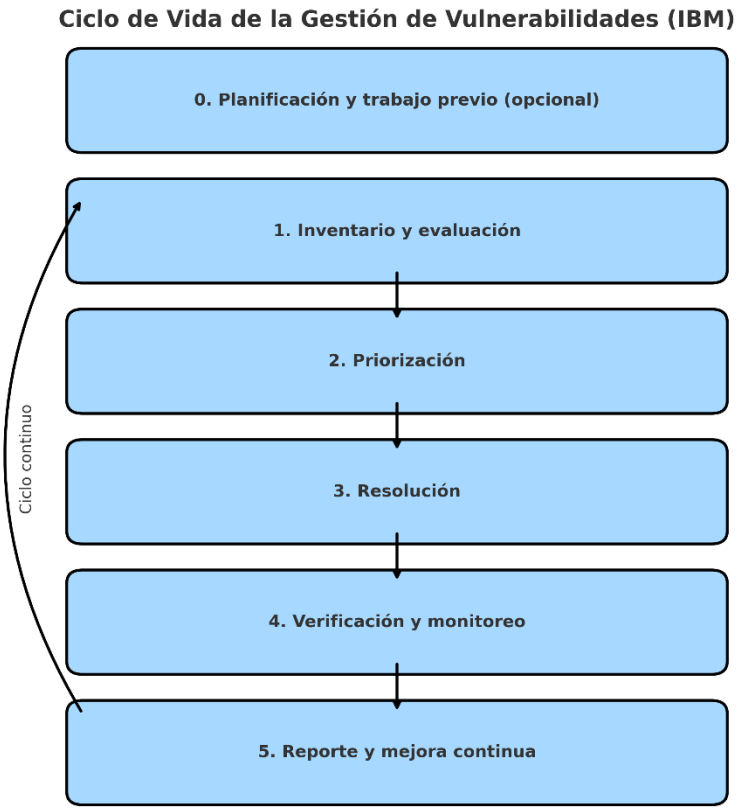
- **Definición:** De acuerdo con INCIBE (Instituto Nacional de Ciberseguridad) una vulnerabilidad es una falla técnica o debilidad de un programa que permite que un atacante acceda a información o datos o de igual manera que lleve a cabo operaciones (acciones) no permitidas a nivel remoto. También es cualquier debilidad **intrínseca** (diseño, arquitectura, configuración) o **introducida** (error humano, implementación, mantenimiento) que pueda ser explotada por una amenaza para dañar un activo.
- **Ciclo de vida de la gestión de vulnerabilidades** (**Para más Información:** https://www.ibm.com/think/topics/vulnerability-management-lifecycle)
    1. **Inventario de Activos y evaluación de vulnerabilidades**
        a. **Definición:** Mantener un inventario actualizado de hardware y software (incluyendo “shadow IT”) y evaluar esos activos con escáneres, pruebas manuales (p. ej., pentesting) e inteligencia de amenazas para descubrir debilidades.
        b. **Ejemplo:** Tu equipo cataloga servidores, laptops y apps internas; luego pasa un escáner que detecta servicios expuestos y configuraciones débiles en una VM de desarrollo.
    2. **Priorización de Vulnerabilidades**
        a. **Definición:** Ordenar qué vulnerabilidades atender primero considerando el puntaje/criticidad externa (CVE/CVSS), la criticidad del activo afectado, el impacto potencial, la probabilidad real de explotación (si hay exploits públicos/activos) y la posibilidad de falsos positivos.
        b. **Ejemplo:** Una vulnerabilidad CVSS 7.5 en un servidor de nómina con datos sensibles se atiende antes que una CVSS 9.0 en un equipo de laboratorio aislado, porque el impacto del primero es mayor.
    3. **Resolución de Vulnerabilidades**
        a. **Definición:** Tratar cada vulnerabilidad priorizada eligiendo entre: remediar (corregir por completo, p. ej. aplicar parches o arreglar una mala configuración), mitigar (reducir la probabilidad/impacto si

no es posible la corrección total) o aceptar (cuando el riesgo es muy bajo o el costo de arreglo es desproporcionado).

   b. **Ejemplo:** Se parchea el servidor web (remediación); para una librería sin parche disponible se añaden WAF/reglas y MFA (mitigación); y se acepta un riesgo menor en una impresora de aula no conectada a datos sensibles (aceptación).
   c. **Nota aclaratoria:** Cuando se habla de **WAF** (Web Application Firewall) estamos hablando sobre en que es un firewall especializado que protege **aplicaciones web** filtrando y monitoreando el tráfico HTTP/HTTPS entre el cliente y el servidor.

4. **Verificación y Monitoreo**
   a. **Definición:** Re-escanear y re-probar para confirmar que la remediación/mitigación funcionó y que no se introdujeron nuevos problemas; además, monitorear continuamente para detectar nuevas vulnerabilidades o controles que quedaron obsoletos.
   b. **Ejemplo:** Tras aplicar parches, el escáner ya no reporta el CVE; se programan verificaciones semanales y alertas para nuevas exposiciones en la red.
5. **Reporte y Mejora**
   a. **Definición:** Documentar lo encontrado, lo que se hizo y los resultados; compartir informes con las partes interesadas y revisar métricas (p. ej., MTTD (Tiempo medio de Detección), MTTR (Tiempo Medio para Responder), número de críticas, recurrencia) para ajustar procesos y mejorar la siguiente iteración.
   b. **Ejemplo:** Un informe mensual muestra reducción del tiempo de parcheo crítico de 14 a 5 días; con base en ello se aumenta la frecuencia de escaneo para servidores expuestos a internet.

a) **Diagrama del Ciclo de Vida de la Gestión de Vulnerabilidades**



- **CVE/CVSS en gestión de vulnerabilidades**
  a) **Información Adicional: https://www.balbix.com/insights/whats-the-difference-between-cve-and-cvss/**
  b) **CVE**
    - **Definición:** Es un identificador único y estandarizado que se asigna a cada vulnerabilidad conocida en software o hardware. Funciona como un “número de serie” para que todos (fabricantes, investigadores, empresas) hablen de la misma vulnerabilidad sin confusión. Lo mantiene la organización **MITRE**, en colaboración con la comunidad. La **MITRE Corporation** es una organización **sin**

fines de lucro en EE. UU. que administra proyectos de interés público, incluyendo bases de datos de ciberseguridad como **CVE**, marcos de ataque como **MITRE ATT&CK**, y trabaja con el gobierno y la industria para mejorar la seguridad tecnológica.

- **Ejemplo:**
  - **CVE-2017-0144 → vulnerabilidad en SMB de Windows, explotada por el ransomware WannaCry.**
  - **CVE-2021-44228 → vulnerabilidad Log4Shell en Apache Log4j.**

c) CVSS

- **Definición:** Es un sistema de puntuación que mide la gravedad de una vulnerabilidad, basado en su impacto y facilidad de explotación. Permite priorizar qué vulnerabilidades atender primero.
- **Escala de CVSS:**
  - **La escala va de 0.0 a 10.0:**
    - 0.0–3.9 → **Bajo**
    - 4.0–6.9 → **Medio**
    - 7.0–8.9 → **Alto**
    - 9.0–10.0 → **Crítico**
- **Ejemplo:**
  - **Log4Shell (CVE-2021-44228) tiene CVSS 10.0, considerado crítico.**
  - **Una fuga de información menor podría tener CVSS 4.0, riesgo moderado.**

- **Ejemplos reales de Vulnerabilidades Famosas**

Año	CVE	Descripción	CVSS
2020	**CVE-2020-1472 (zerologon)**	Fallo crítico en protocolo Netlogon de Windows Server que permite tomar control de un dominio.	**10.0 (Crítico)**

2021	**CVE-2021-44228 (Log4Shell)**	Vulnerabilidad en Apache Log4j que permite ejecución remota de código (RCE).	**10.0 (Crítico)**
2022	**CVE-2022-30190 (Follina)**	Vulnerabilidad en Microsoft MSDT explotable mediante documentos de Office.	**7.8 (Alto)**
2023	**CVE-2023-34362 (MOVEit Transfer)**	Fallo en software de transferencia de archivos explotado en ataques de ransomware masivos.	**9.8 (Crítico)**
2024	**CVE-2024-3094(XZ Utils backdoor)**	Puerta trasera introducida maliciosamente en la librería de compresión XZ, afectando Linux.	**10.0 (Crítico)**

- **Diferencia clave frente a amenaza y riesgo**
    a) *Amenaza*: quién/qué puede explotar la debilidad. Que vulnerabilidad se puede aprovechar para causar daño.
        - **Ejemplo:** Un atacante que lanza un ataque de ransomware contra una red hospitalaria.
    b) *Riesgo*: probabilidad e impacto cuando la amenaza activa la vulnerabilidad. (**Probabilidad:** qué tan posible es que una amenaza explote una vulnerabilidad. **Impacto:** el daño o consecuencia si la amenaza se materializa.)
        - **Ejemplo:** Si un hospital no aplica parches de seguridad y la probabilidad de ransomware es alta, el riesgo es que los sistemas médicos queden inutilizables, afectando pacientes (impacto crítico).
- **Clasificaciones habituales**
    a) **Inherentes** vs. **contextuales** (dependen de la configuración o del entorno).
    b) **Latentes** (aún no descubiertas) vs. **Activas** (con exploit público).

Vulnerabilidades en hardware, software o datos

A. Hardware

- **Side-Channel Attacks (**Ataques de canal lateral**)**
    - **Definición:** Aprovechan la ejecución especulativa y paralela de las CPU para filtrar información (tiempos de acceso a memoria, consumo eléctrico, radiación electromagnética). Es como **adivinar una contraseña escuchando los clics del teclado**: no atacas la contraseña directamente, sino el "ruido" que produce.
    - **Ejemplos:**
        - **Spectre y Meltdown (2018):** filtración de memoria protegida del kernel.
        - **Foreshadow (L1TF, 2018):** ataques al caché **L1** en Intel.
        - **ZombieLoad (2019):** extrae datos de la cola de carga de la CPU.
    - **Mitigación:** parches de microcódigo y rediseño de arquitecturas de CPU.
- **Rowhammer**:
    - **Definición**: Manipula celdas de memoria DRAM golpeando repetidamente filas adyacentes para provocar **flips de bits**.
    - **Ejemplo**: Un atacante puede modificar privilegios en memoria sin necesidad de vulnerabilidad de software.
    - **Mitigación**: uso de memoria ECC y técnicas TRR (*Target Row Refresh*).
- **Firmwares inseguros (BIOS/UEFI)**:
    - **Definición**: Cuando el firmware carece de **firmas digitales seguras**, puede ser modificado para incluir **implantes persistentes**. Basicamente es como **un ladrón que cambia la cerradura de tu casa**: aunque cambies las ventanas (SO), el acceso raíz ya está comprometido. Entiéndase que **BIOS**/**UEFI** nos referimos a los dos tipos de firmware de arranque que inicializan el hardware antes de cargar el sistema operativo. **UEFI (Unified Extensible Firmware Interface)** es una evolución del BIOS

    - **Ejemplo**: El malware **LoJax (2018)** infectó UEFI para sobrevivir a formateos y reinstalaciones de SO.
    - **Mitigación**: activar **Secure Boot**, actualizar BIOS/UEFI solo desde fuentes confiables.
- **Supply-Chain Tampering (Manipulación en la cadena de suministros)**:
    - **Definición**: Alteración de componentes **antes de llegar al usuario**, ya sea en fábrica, transporte o instalación.
    - **Ejemplo**: Casos reportados de **chips espía insertados en placas base** o backdoors en routers.
    - **Mitigación**: certificaciones de proveedores, auditorías y validación de hardware antes de su uso.

B. Software

- **Memory Safety**:
    - **Definición**: Errores al manejar memoria en C/C++ permiten lecturas/escrituras fuera de límites.
    - **Ejemplos**:
        - **Buffer overflow:** escribir más datos de los permitidos → ejecución de código arbitrario.
        - **Use-after-free:** acceder a memoria ya liberada.
        - **Double free:** liberar memoria dos veces → corrupción.
    - **Mitigación:** Uso de lenguajes seguros (Rust, Java), ASLR (*Address Space Layout Randomization*), validación estricta.
- **Injection**:
    - **Definición**: Entrada maliciosa se interpreta como comando o consulta.
    - Tipos:

1. SQL Injection: robar/alterar datos en base de datos.
2. LDAP Injection: manipular directorios corporativos.
3. Command Injection: ejecutar comandos en el SO.
4. NoSQL Injection: explotar bases modernas (MongoDB).

- **Mitigación:** Declaraciones Preparadas, validación de entrada, WAF.

- **Auth/Z-Session (Autenticación, Autorización y Sesiones)**:
  - **Definición**: Fallas en gestión de identidad y permisos.
  - **Ejemplos:**
    - **CSRF (Cross-Site Request Forgery):** fuerza a un usuario autenticado a ejecutar acciones sin querer.
    - **IDOR (Insecure Direct Object Reference):** acceder a recursos cambiando un ID en la URL.
    - **JWT mal configurado:** firmas débiles o sin expiración.
  - **Caso real:** Aplicaciones web que permiten ver datos de otros usuarios cambiando un parámetro *user_id*.
  - **Mitigación:** MFA, tokens con expiración, control de acceso basado en roles (RBAC).
- **Insecure Deserialization (Deserialización Insegura)**:
  - **Definición**: Manipulación de objetos serializados para ejecutar código arbitrario al deserializarlos.
    - Entiéndase que **Serializar** es cuando se toma un objeto y se convierte en un formato que pueda ser almacenado, sin embargo Deserializar es tomar ese archivo, serializado, o cadena (JSON, XML, binario) y reconstruir el objeto original en memoria.
  - **Ejemplo:** Cargar un objeto Java malicioso en un servidor vulnerable → RCE (ejecución remota de código).
  - **Mitigación:** Validar tipos de objetos, usar formatos seguros (JSON), evitar deserializar datos no confiables.
- **Dependency Issues**:
  - **Definición**: Uso de librerías externas vulnerables o desactualizadas.

- **Ejemplos**:
  - **Log4Shell (CVE-2021-44228):** vulnerabilidad crítica en Apache Log4j → RCE masivo.
  - Paquetes NPM infectados en ataques de supply chain.
  - **Mitigación:** Escaneo de dependencias (SCA – *Software Composition Analysis*), actualizaciones regulares, SBOM (*Software Bill of Materials*).

C. Datos

- **Exposición pasiva**:
  - **Definición**: Cuando los datos quedan **abiertos sin protección**, accesibles para cualquiera en internet.
  - **Ejemplos**:
    - Ejemplos:
    - Buckets de Amazon S3 sin autenticación.
    - Servidores FTP anónimos con acceso libre.
  - **Mitigación:** Revisar configuraciones de nube, aplicar principio de mínimo privilegio (**(PoLP, Principle of Least Privilege)** establece que **todo usuario, proceso o sistema debe tener únicamente los permisos estrictamente necesarios para realizar sus funciones** y nada más), escaneo automático de buckets expuestos.
- **Meta-data leakage (Fuga de Metadatos)**:
  - **Definición**: Información sensible oculta en documentos, imágenes o archivos que **revela más de lo debido**.
  - **Ejemplos:**

        - Documentos Word con nombres de usuarios internos y rutas de red.
        - Fotos con GPS EXIF data, revelando ubicación.
    - **Caso real:** Soldados revelaron la ubicación de bases militares en 2018 por apps de fitness (Strava) que guardaban GPS.
- **Criptografía débil/ausente**:
    - **Definición**: Cuando los datos no se protegen con cifrado moderno, quedando expuestos en caso de robo.
    - **Ejemplos:**
        - Bases de datos almacenadas en texto plano ("at rest").
        - Uso de algoritmos rotos (MD5, DES).
        - Claves incrustadas en el código fuente.
    - **Mitigación:** Cifrado fuerte (AES-256, TLS 1.3), gestión de claves segura (HSM, Vault).
- **Retención excesiva**:
    - **Definición**: Guardar datos **más tiempo del necesario**, aumentando la superficie de ataque.
    - **Ejemplos:**
        - Empresas que almacenan historiales de clientes por años sin razón legal.
        - Logs de sistemas con credenciales antiguas.
    - **Caso real:** Varias multas bajo GDPR a empresas por no eliminar datos después del tiempo estipulado.
    - **Mitigación:** Políticas de data retention, aplicar "right to be forgotten" (GDPR), depuración automática de datos.
        - **Right to be Forgotten (Derecho al Olvido):** Visto en el articulo 17 de la normativa GDPR en que es el derecho de una persona a

solicitar que una organización **elimine sus datos personales** cuando ya no son necesarios, fueron tratados de forma ilegal o cuando el interesado retira su consentimiento. **Para mas información: https://eur-lex.europa.eu/eli/reg/2016/679/oj**

Ejemplo transversal: Un contenedor Docker público contiene .env con claves (categoría datos) y la imagen arrastra una librería vulnerable (software); si se despliega en un clúster sin TPM, surge una debilidad de firmware (hardware).

Vulnerabilidades en lógica, algoritmos, procesos o protocolos

- **Fallos algorítmicos**
    - Los **fallos algorítmicos** son vulnerabilidades que ocurren no por errores de programación básicos, sino por cómo está diseñado un **algoritmo, proceso o protocolo**. Pueden comprometer rendimiento, disponibilidad e incluso integridad de datos.
    - *Complexity Bombs (ReDoS – Regular Expression Denial of Service)*: Una expresión regular mal diseñada (con retro referencias, repeticiones o patrones ambiguos) puede causar que el motor de regex entre en **backtracking excesivo**, consumiendo CPU de manera desproporcionada.
        - **Ejemplo:**
            - **Regex vulnerable:** (a+)+$
            - **Entrada maliciosa:** "aaaaaaaaaaaaaaa!"
            - El motor evalúa miles de combinaciones antes de fallar → consumo de CPU → **Denegación de Servicio (DoS)**.
        - **Caso real:** Varias librerías en JavaScript (ej. moment.js, npm packages) han sufrido vulnerabilidades **ReDoS** reportadas en CVE.
        - **Mitigación:**
            - Evitar regex con retro referencias costosas.
            - Usar *timeouts* en evaluaciones de regex.
            - Revisar librerías externas que usen regex.

        - **Analogía:** Es como una cerradura mal diseñada que al probar demasiadas combinaciones se atasca y bloquea la puerta.
    - *Integer Overflow (Desbordamiento de Enteros)*:
        - **Definición:** Sucede cuando un valor numérico excede el máximo que puede almacenar su tipo de dato, “dando la vuelta” al contador y reiniciando en cero o en valores negativos.
        - **Ejemplo:**
            - contadores de likes en redes sociales que vuelven a cero tras 4 294 967 296 eventos.
        - **Casos reales:**
            - Bugs en videojuegos donde las puntuaciones o vidas vuelven a cero al alcanzar el límite.
            - En seguridad, el integer overflow puede usarse para evadir controles de acceso o provocar fallos en memoria (ej. ataques a librerías en C).
        - **Mitigación:**
            - Usar tipos de datos más grandes (uint64).
            - Validar entradas numéricas antes de procesarlas.
            - Lenguajes modernos como Rust y Python lanzan excepciones al detectar overflow.
- **Errores de proceso / lógica de negocios**
    - ***Mediación incompleta*:**
        - **Definición:** Cuando el sistema valida una condición en una etapa, pero no la vuelve a validar en las etapas críticas posteriores.
        - **Ejemplo:** En una tienda en línea, el precio se valida al agregar al carrito, pero no al procesar el pago. Un atacante puede manipular el parámetro price=10 en lugar de 1000 en la solicitud final → compra con descuento fraudulento.
        - **Mitigación:** Validar precios y permisos en cada transacción crítica del servidor, no solo en el cliente o en pasos iniciales.
    - ***Race Condition / TOCTOU (Time of Check, Time of Use)*:**

        - **Definición:** Un error cuando la **verificación** y el **uso de un recurso** ocurren en momentos distintos, permitiendo que un atacante manipule el estado entre ambos pasos.
        - **Ejemplo:** Un programa verifica que un usuario tiene permiso para escribir en un archivo temporal (check), pero antes de usarlo (use), el atacante reemplaza el archivo con un enlace simbólico hacia /etc/passwd. El sistema escribe allí con privilegios elevados.
        - **Mitigación:** Usar operaciones atómicas (verificación y uso en una sola acción), y manejar archivos temporales con identificadores únicos y seguros.
- **Defectos de protocolo**
    - *Downgrade*
        - **Definición:** Ataque que fuerza al servidor y cliente a usar una versión más débil del protocolo (ej. SSL 3.0 en lugar de TLS 1.2).
        - **Ejemplo real:** El ataque POODLE (2014) explotaba SSL 3.0, permitiendo descifrar cookies de sesión. **Mas Información:** **https://www.cisa.gov/news-events/alerts/2014/10/17/ssl-30-protocol-vulnerability-and-poodle-attack**
        - **Mitigación:** Deshabilitar protocolos inseguros (SSL, TLS < 1.2) y aplicar secure fallback.
        - **Entiendase que cuando hablamos de SSL (Secure Sockets Layer) y TLS (Transport Layer Security),** donde **SSL** fue el primer protocolo de cifrado (1990) a establecerse para cifrar comunicaciones en internet y **TLS** es el sucesor de SSL en el cual fue diseñado por la IETF en 1999 para corregir fallos de SSL.

    - *Clear-text protocols*:
        - **Definición:** Protocolos que transmiten datos **sin cifrado**, lo que expone credenciales y mensajes a sniffing.
        - **Ejemplos:**
            - **Telnet** (contraseñas en claro).

        - **FTP** sin TLS.
        - **SMTP** sin STARTTLS → correos interceptados.
    - **Mitigación:** Sustituir por protocolos cifrados (**SSH, SFTP, FTPS, SMTPS**).

- *Weak Handshake*:
- **Definición**: Ataque al handshake de Bluetooth BR/EDR, donde un man-in-the-middle negocia que ambas partes usen claves de solo 1 byte aleatoriamente.
- **Ejemplo real:** El KNOB Attack (2019) permitía descifrar comunicaciones Bluetooth y espiar tráfico entre dispositivos.
- **Mitigación:** Aplicar parches de fabricantes, usar Bluetooth con versiones actualizadas y evitar emparejar en entornos no seguros.

- **Impacto típico**
    - **Transacciones alteradas**
        - **Definición:** Cuando un atacante modifica operaciones en curso (pagos, transferencias).
        - **Ejemplo:** Cambiar el monto en una transacción bancaria en línea.
        - **Mitigación:** Validación en servidor, firmas digitales en transacciones.
        - **Nota Aclaratoria:** Cuando se habla de Firmas Digitales nos referimos a **mecanismos criptográficos** que proveen Autenticidad, Integridad y No Repudio (el remitente no puede negar que ha enviado) de datos.
    - **Escalada de privilegios**
        - **Definición:** Obtener permisos mayores a los autorizados.
        - **Ejemplo:** Un usuario normal accede a funciones de administrador por un fallo de control de acceso.
        - **Mitigación:** RBAC (control basado en roles), pruebas de autorización en cada acción.

- **Fuga de sesión**
  - **Definición:** Robo o exposición del identificador de sesión de un usuario.
  - **Ejemplo:** Un atacante roba la cookie de sesión en un Wi-Fi público y toma control de la cuenta.
  - **Mitigación:** Cookies seguras (HttpOnly, Secure), TLS, expiración rápida de sesiones.
- **Robo de tokens OAuth**
  - **Definición:** Captura indebida de tokens de autorización usados para acceder a APIs y servicios.
  - **Ejemplo:** Un atacante roba un token de acceso de Google OAuth y accede a Gmail sin contraseña.
  - **Mitigación:** Rotación frecuente de tokens, scopes limitados, MFA, almacenamiento seguro.
  - ***Entiendase que cuando se habla de OAUTH nos referimos a Open Authorization** que es un **estándar abierto** que permite a las aplicaciones obtener acceso limitado a recursos de un usuario **sin necesidad de compartir sus credenciales (usuario/contraseña)**.*

Vulnerabilidades en usuarios o personal

- **Factores Cognitivos**
  - *Password Fatigue*:
    - **Definición:** Cansancio por manejar múltiples contraseñas → usuarios reutilizan o simplifican claves.
    - **Ejemplo:** Misma contraseña usada en correo, redes sociales y banca → ataques de credential stuffing.
    - **Mitigación:** Uso de gestores de contraseñas y MFA.
    - *Atajos* (heurísticas) que llevan a saltarse validaciones de 2FA.
  - **Atajos en 2FA**

        - **Definición:** Usuarios buscan simplificar procesos, evitando pasos de seguridad.
        - **Ejemplo:** Guardar tokens de 2FA en notas del móvil o aceptar notificaciones push sin leer.
        - **Mitigación:** Educación en riesgos, reforzar **autenticación adaptativa** (solo pedir 2FA en entornos sospechosos).
- **Ingeniería Social Avanzada**
    - *Vishing* (voice phishing):
        - **Definición:** Engaño vía llamadas telefónicas.
        - **Ejemplo:** Un "soporte TI" llama pidiendo resetear la contraseña corporativa.
        - **Mitigación:** Políticas claras: TI nunca pide contraseñas por teléfono, verificar identidad por otro canal.

    - *Deepfake Phishing*:
        - **Definición:** Uso de audio/vídeo sintético para suplantar directivos.
        - **Ejemplo:** En 2023, fraude bancario donde voz falsa de un "CEO" ordenó transferencias (WSJ).
        - **Mitigación:** Doble validación de transacciones críticas, incluso si vienen de ejecutivos.

    - *Quid pro quo*:
        - **Definición:** Engaño ofreciendo algo a cambio.
        - **Ejemplo:** Entregar un USB de "regalo corporativo" que en realidad instala malware.
        - **Mitigación:** Política de dispositivos externos, escaneo antes de conectar hardware.

- **Insider Threat**
    - **Malicioso**: Empleado con acceso legítimo abusa de sus privilegios.

        - **Mitigación: Principio de mínimo privilegio**, monitoreo de accesos.
    - **No Malicioso**: borrado accidental de logs críticos.
        - **Backups frecuentes**, automatizar retención de logs.

- **Contramedidas centradas en el usuario**
    - **Definición:** Medidas de seguridad diseñadas para fortalecer al usuario como primera línea de defensa.
    - **Ejemplo:** Simulaciones de phishing, revisiones periódicas de privilegios.
    - **Mitigación:**
        - Formación continua en ciberseguridad.
        - Phishing drills regulares.
        - Aplicar least privilege en accesos.
        - Revisiones trimestrales de permisos y cuentas inactivas.

Comparar fallos lógicos y humanos

Criterio	Fallo lógico	Fallo humano
Origen	Diseño, algoritmo, código	Comportamiento, toma de decisiones
Mitigación	Revisiones de código, pruebas automatizadas, principles of secure design	Concienciación, capacitación, políticas y monitoreo

Criterio	Fallo lógico	Fallo humano
Detección	Herramientas estáticas/dinámicas, auditorías	Simulaciones de phishing, análisis forense de logs
Ejemplo	Uso de entero de 32 bits para conteo de visitas (overflow)	Contraseña “123456” guardada en nota adhesiva

Análisis de impacto (CIA + CVSS)

1. **Triada CIA**
    - *Confidencialidad*: ¿Puede un atacante leer información privada?
    - *Integridad*: ¿Puede modificarla o corromperla?
    - *Disponibilidad*: ¿Puede impedir el acceso legítimo?
2. **Métricas CVSS v3.1 relevantes**
    - **Attack Vector (AV)**: física, local, adyacente, red. Significa que puede explotarse **remotamente, a través de internet**.
    - **Attack Complexity (AC)**: baja o alta (¿requisitos de entorno?). La explotación **no necesita condiciones especiales**.
    - **Privileges Required (PR)** y **User Interaction (UI)**. El atacante **no necesita credenciales** previas. **UI** - El ataque **no requiere que la víctima haga clic ni ejecute nada**.
    - **Scope (S)**: ¿puede afectar a otros componentes? El fallo permite que el atacante **salga del componente afectado y comprometa otros sistemas**.
    - **Impact Metrics (C, I, A)**: ninguno, bajo, alto.
        1. Confidencialidad: se puede robar cualquier dato.
        2. Integridad: el atacante puede modificar datos/sistema.
        3. Disponibilidad: puede tumbar el servicio o instalar ransomware.
3. **Ejemplo de Scoring** – *Log4Shell* (CVE-2021-44228)
    - AV: Red (0.85)
    - AC: Baja (0.77)

    - PR: Ninguno (0.85)
    - UI: Ninguno (0.85)
    - S: Cambiado (1.00)
    - C=I=A: Alto (0.56 cada uno)
    - **CVSS Base Score ≈ 10.0 (Crítico)**
4. **Matriz de riesgo cualitativa**
    - Eje X: probabilidad (Baja–Alta)
    - Eje Y: impacto (Bajo–Crítico)
    - *Log4Shell* se sitúa en la esquina superior derecha → máxima prioridad de remediación.
5. **Priorización de parches**
    - *Critical* (>9.0): corregir en < 24 h.
    - *High* (7.0–8.9): < 72 h.
    - *Medium* (4.0–6.9): próximos ciclos de mantenimiento.
    - *Low* (<4.0): documentar y monitorizar.

Relación con Otros Conceptos

- La clasificación de vulnerabilidades se conecta con **Riesgos/amenazas/daños** (siguiente lección del Módulo B).
- Los fallos lógicos enlazan con las **vulnerabilidades de software** (Módulo D) y los controles de programación defensiva (Módulo C).
- Las debilidades humanas anticipan los **aspectos sociales y ético-legales** (Módulo G).

Resumen de la Lección

Exploramos la anatomía de las vulnerabilidades clasificándolas en tres categorías principales: técnicas en hardware/software/datos, de diseño en lógica y protocolos, y humanas en el comportamiento del personal. Aprendimos a diferenciarlas y a evaluar su impacto mediante la triada CIA y el CVSS. Esta comprensión es esencial para priorizar defensas y reducir riesgos de forma efectiva.

Actividad de la Lección

Laboratorio de clasificación de vulnerabilidades (90 min):

1. El profesor entrega cinco descripciones de incidentes reales (p. ej., *Heartbleed*, *Freak*, *PhishMe*).
2. Cada equipo debe:
   a. Identificar la categoría de vulnerabilidad (técnica, lógica o humana).
   b. Mapear el impacto a CIA.
   c. Asignar un rango aproximado de CVSS (bajo, medio, alto, crítico).
3. Presentar un póster digital (1 diapositiva) que justifique la clasificación.
   Entrega: archivo PDF al LMS.
   Criterios de evaluación: precisión técnica (50 %), claridad de la justificación (30 %), creatividad visual (20 %).

Referencias Adicionales

- Balbix. (2023, 3 de abril). *What's the difference between CVE and CVSS?* Balbix. https://www.balbix.com/insights/whats-the-difference-between-cve-and-cvss/
- Common Vulnerabilities and Exposures (CVE). (2025). *CVE-2025-43300*. CVE. https://www.cve.org/CVERecord?id=CVE-2025-43300
- Cybersecurity and Infrastructure Security Agency (CISA). (2014, 17 de octubre). *SSL 3.0 protocol vulnerability and POODLE attack*. CISA. https://www.cisa.gov/news-events/alerts/2014/10/17/ssl-30-protocol-vulnerability-and-poodle-attack
- Cybersecurity and Infrastructure Security Agency (CISA). (s. f.). *Known exploited vulnerabilities catalog*. CISA. https://www.cisa.gov/known-exploited-vulnerabilities-catalog
- European Union. (2016). *Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016 relativo a la protección de las personas*

físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos (GDPR). EUR-Lex. https://eur-lex.europa.eu/eli/reg/2016/679/oj

- IBM. (2023, 28 de julio). Vulnerability management lifecycle. IBM Think. https://www.ibm.com/think/topics/vulnerability-management-lifecycle

- Pfleeger, C. P., Pfleeger, S. L., & Coles-Kemp, L. (2023). *Security in Computing* (6.ª ed.). Addison-Wesley.
- Bishop, M. (2018). *Computer Security: Art and Science* (2.ª ed.). Addison-Wesley.
- Du, W. (2022). *Computer Security: A Hands-on Approach* (3.ª ed.).
- MITRE. *Common Vulnerabilities and Exposures (CVE) List*.
- FIRST.org. *Common Vulnerability Scoring System v3.1: Specification Document*.