Repaso para Examen 2 Final

Cibersecurity COMP2700

1. El principio de **Privacy by Design** significa que la protección de datos y la minimización de la información se consideran desde el diseño inicial del sistema y durante todo su ciclo de vida.

2. Un ataque de **ransomware** cifra los archivos de la víctima y luego exige un pago para restaurar el acceso a esos datos.

3. Una **inyección SQL** ocurre cuando la aplicación construye consultas concatenando directamente datos que vienen del usuario sin validarlos ni parametrizarlos.

4. Es falso decir que un ataque de **phishing** siempre depende de malware complejo; en realidad se basa principalmente en la **ingeniería social** para engañar al usuario.

5. Un **desbordamiento de búfer (buffer overflow)** ocurre cuando un programa escribe más datos de los que caben en una zona de memoria reservada, sobrescribiendo direcciones o variables cercanas.

6. En esquemas de cifrado para bases de datos, la **CEK/DEK** cifra directamente los datos sensibles y la **CMK/KEK** se usa para cifrar y proteger esas CEK/DEK, normalmente almacenadas en un KMS o HSM.

7. La **Row-Level Security (RLS)** sirve para controlar qué filas específicas de una tabla puede ver o modificar cada usuario, sin necesidad de duplicar tablas.

8. El **GDPR** y la **Ley 111-2022 de Puerto Rico** tienen como propósito proteger la privacidad de las personas y obligar a las organizaciones a

manejar responsablemente los datos, incluyendo notificación de incidentes y posibles multas.

9. El desarrollo profesional en ciberseguridad requiere **aprendizaje continuo**, mediante certificaciones, cursos en línea y laboratorios prácticos, para no quedar obsoleto ante nuevas amenazas.

10. Un ejemplo de problema de **accesibilidad** que también se convierte en riesgo de seguridad es usar un **CAPTCHA solo visual**, que impide que personas ciegas puedan autenticarse con un lector de pantalla.

11. Es falso que la validación de entradas deba hacerse únicamente en la base de datos; se debe validar también en los **endpoints y en la frontera del sistema** para fortalecer la seguridad.

12. Una buena **gestión de secretos** evita guardar contraseñas o claves API en el código fuente y prefiere almacenarlas cifradas en servicios especializados como gestores de secretos.

13. En un **pipeline de CI/CD seguro** se integran herramientas como *Bandit* o *pip-audit* para detectar vulnerabilidades en el código y sus dependencias antes de desplegar a producción.

14. El principio de **"sospecha mutua"** entre microservicios indica que cada servicio debe validar autenticación, autorización y formato de los datos que recibe, aun cuando provengan de otros servicios internos.

15. El **sandboxing** busca aislar procesos con permisos limitados para que, si se comprometen, el impacto y el daño queden contenidos en un entorno controlado.

16. Para fortalecer **HTTPS con TLS 1.3** se deben usar solo protocolos modernos, suites de cifrado fuertes y mecanismos para renovar automáticamente los certificados.

17. La diferencia clave es que un **IDS** se enfoca en detectar y alertar sobre actividad sospechosa, mientras que un **IPS** además puede bloquear el tráfico malicioso en tiempo real.

18. El uso de **Argon2id** junto con sal (salt) y pepper permite almacenar contraseñas de forma más segura y resistente a ataques con GPU y diccionarios.

19. **Friendly-Captcha** reemplaza CAPTCHAs tradicionales usando pruebas criptográficas automáticas en el navegador, mejorando la accesibilidad y la privacidad del usuario.

20. Según **PCI DSS**, el código de seguridad de la tarjeta (**CVV/CVC**) no debe almacenarse nunca después de completarse la autorización del pago.

21. Las **vistas agregadas** que solo muestran SUM, AVG, COUNT u otras funciones ayudan a entregar valor analítico sin exponer registros individuales, apoyando la privacidad.

22. Cuando un ataque de **ransomware** afecta un hospital y retrasa cirugías u otros servicios, el impacto es principalmente social y puede comprometer directamente la vida y la salud de las personas.

23. El **fuzzing** consiste en enviar a un sistema gran cantidad de entradas aleatorias o malformadas para descubrir fallos y vulnerabilidades que no se detectan con pruebas normales.

24. Una buena gestión de **vulnerabilidades y parches** incluye aplicar pruebas de regresión y usar estrategias como **canary releases** para desplegar cambios primero a un grupo pequeño de usuarios y reducir el riesgo.

25. El principio de **Zero Trust** en redes modernas se resume como “nunca confiar, siempre verificar”; cada solicitud debe autenticarse y autorizarse, incluso dentro de la red interna.

26. En un esquema de **clasificación de información**, la categoría **“Restricted”** suele reservarse para información altamente sensible que requiere controles de acceso muy estrictos.

27. Una forma de **inyección en NoSQL** es enviar un JSON malicioso como {"$ne": null} para manipular consultas en bases de datos como MongoDB y devolver más documentos de los esperados.

28. Los **honeypots** son sistemas señuelo diseñados para atraer atacantes, registrar sus acciones y ayudar a analizar su comportamiento sin exponer los sistemas productivos reales.

29. Una buena práctica al diseñar **ACL o reglas de firewall** es documentar cada regla, registrar los intentos de tráfico denegado y revisar periódicamente las listas de acceso.

30. El modelo **CIA** evalúa el impacto de una vulnerabilidad considerando la **Confidencialidad, Integridad y Disponibilidad** de la información y de los servicios afectados.