Alfonso Rios. 8.75

Fundamentos y Técnicas de Seguridad para Aplicaciones

Evaluación de fin de cursado

1. *Gestión de Dependencias como Problema de Seguridad* — 2 puntos

Analiza por qué la gestión de dependencias de software (ej. en NPM, PyPI, apt) se ha convertido en un vector crítico de ataques a la cadena de suministro. Propón dos prácticas, una técnica y una organizacional, para concientizar o reducir este riesgo.

2. *Antipatrones de seguridad* — 1 punto

¿Cuáles de los siguientes son *antipatrones de seguridad* reconocidos? Indica cuáles son, e indica qué correspondería hacer si encuentras código con este *antipatrón* en un sistema que estés trabajando.

a) "Security by Obscurity" (Seguridad por Oscuridad).
b) Almacenar contraseñas en texto plano en la base de datos.
c) Implementar el principio de mínimo privilegio para los servicios.
d) Realizar validación de entrada únicamente del lado del cliente.

3. *Desbordamientos* — 2 puntos

Si bien los desbordamientos más famosos (¡y por buenas razones!) son los desbordamientos de pila (*stack overflows*), la cantidad de amenazas que pueden caracterizarse de esta manera es mucho mayor. Compara la explotación de un *desbordamiento de pila* (*stack overflow*) con un *desbordamiento de enteros* (*integer overflow*).

4. *Enumeración de debilidades, enumeración de vulnerabilidades* — 2 puntos

Dos de los índices mas importantes para la comprensión y el seguimiento de la seguridad en aplicaciones son el **CWE** (*Common Weakness Enumeration*) y el **CVE** (*Common Vulnerabilities Enumeration*). ¿Cuál es la relación entre ambos? ¿Existe algún *camino* que lleve a algún *problema* que descubramos en determinado código a *transitar* de clasificarse como CWE a clasificarse como CVE (o a la inversa)?

5. *Ataques de inyección* — 2 puntos

Elige un ataque de inyección específico (SQLi, des-serialización, SSRF, etc.) y propón una estrategia de "defensa en profundidad", ubicando controles de seguridad distintos en diferentes capas (ej. aplicación, red, SO) para mitigarlo.

6. *Tipos de inyección* — 1 punto

Relaciona cada concepto (*tipo de inyección*) con su descripción correspondiente.

Tipo de inyección	Descripción
a) Inyección de comandos	*a*) La aplicación des-serializa datos no confiables, instanciando objetos o invocando métodos arbitrarios
b) Inyección de objetos	*b*) Datos no validados se envían a un intérprete de comandos del sistema operativo
c) Server-Side Request Forgery (SSRF)	*c*) Entradas de usuario se incorporan directamente en una consulta de base de datos, alterando su lógica
d) Inyección SQL	*d*) La aplicación es engañada para realizar consultas a un destino interno o restringido

a - b
b - a
c - d
d - c

Alfonso Rios

1 Aunque los gestores de dependencias son muy cómodos por su facilidad es un problema de seguridad ya que muchos sitios web que utilizan node/Js/Ts pueden ser vulnerados desde un único punto con npm o cualquier software escrito en Python. La práctica técnica puede ser mantener las bibliotecas lo más actualizadas posible, a pesar de que este sea el vector de ataque, una vez identificado y parchado tenemos que asegurar que ya tenemos la solución instalada. Organizacionalmente, implementar políticas que aseguren que los paquetes instalados sean auditados cada cierto tiempo. Terminamos, con ambas opciones, muy cerca de donde comenzamos ¿Cómo?
¿Y qué me dices de (a)?

2. b) Contraseñas en texto plano: Si esto es encontrado, yo buscaría en caso de tener un servidor o servicio de manejo de contraseñas permitido, utilizarlo, de necesitarse algo más accesible, un servicio de gestión de secretos y si necesariamente se necesitan en una BD las contraseñas, pasarlas por una encripción donde la llave pueda almacenarse en un gestor de secretos. La respuesta general es que si crees que debes almacenar una contraseña, lo pienses bien de vuelta. ¡Guarda un hash!

d) En casos específicos el mismo input del cliente puede hacer que la validación sea evitada y esto deja a nuestro backend expuesto si me encontrara esto, validaría todos los datos en el backend así agregando otra capa de sanitización para todos los datos ingresados por un cliente a nuestro back.

3. El desbordamiento de pila consiste en lograr que un programa use memoria fuera de la pila? destinada a el y el de enteros ocurre cuando el número entero (por lo general resultado de una operación aritmética) supera el valor máximo que se puede representar con los bits destinados a un entero.
memoria que no quepa en determinado buffer que reside en el área de pila (stack) del proceso. Esto puede llevar a la sobreescritura de dirección de retorno, y secuestro de la ejecución del proceso.

4. La relación entre ambos se puede decir que es de cause-efect ya que el CWE se enfoca en el "porque" o la causa y el CVE se enfoca en el "síntoma". Por ejemplo un CWE puede ser una inyección de SQL como método y el CVE es una vulnerabilidad específica en algún software.

El camino para clasificarlos como uno o el otro es difícil ya que al tratarse de conceptos distintos es complicado definirlo pero en general un CWE nuevo podría ser solo si no se encuentre una debilidad con esas características y el CVE es un ejemplo en vivo de alguna debilidad explotada.

5 Un SQLi, desde la capa de application de frente al usuario se tiene que hacer una limpieza de todos los inputs validando signos colocados en lugares donde no se debería, a nivel backend se debe validar el tipado del regreso de información de DB, es decir, si el Query debe regresar una lista de enteros validar que sí lo sean. También monitorear el tiempo que tarda en correr el query, Si sabemos que un query no debe tomar más de 30 segundos, si los excede, devolver un timeout. Esta es una estrategia efectiva además de todas las limpiezas que ya existen y que los frameworks hacen por nosotros. A nivel "defensa en profundidad" me quedé esperando que exploraras algunas otras capas o revisiones...

6. a-b
   b-a
   c-d
   d-c