



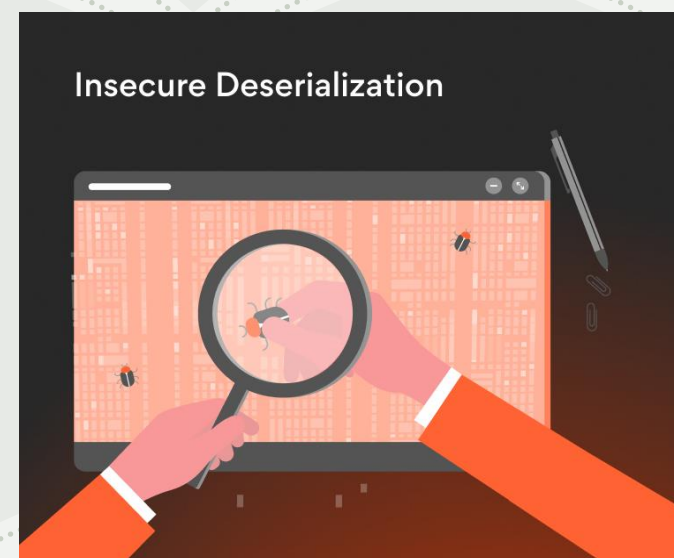
Deserialización de datos no confiables

ALUMNO

RODOLFO GIRÓN CORONEL

¿Qué es la deserialización?

LA DESERIALIZACIÓN ES EL PROCESO DE CONVERTIR DATOS EN UN FORMATO SERIALIZADO (COMO JSON O XML) DE NUEVO A UN OBJETO O ESTRUCTURA DE DATOS. ESTA OPERACIÓN ES COMÚN EN APLICACIONES QUE UTILIZAN LA COMUNICACIÓN ENTRE DIFERENTES COMPONENTES O SISTEMAS. SIN EMBARGO, CUANDO SE MANEJA DE MANERA INSEGURA, LA DESERIALIZACIÓN PUEDE CONVERTIRSE EN UNA VULNERABILIDAD.



La vulnerabilidad

Ocurre cuando una aplicación deserializa datos provenientes de fuentes no confiables sin validar adecuadamente esa entrada.

Esto puede ser explotado por un atacante para ejecutar código malicioso en el sistema objetivo.



Ejemplo demostrativo

Este fragmento de código deserializa un objeto de un archivo y lo utiliza como botón de la interfaz de usuario: Este código no intenta verificar el origen o el contenido del archivo antes de deserializarlo. Un atacante puede reemplazar el archivo deseado con un archivo que contenga código malicioso arbitrario que se ejecutará cuando se presione el botón.

Para mitigar esto, defina explícitamente `readObject()` final (El método `readObject` lee los objetos del flujo de entrada, en el mismo orden en el que ha sido escritos.) para evitar la deserialización. Un ejemplo de esto es:

Lenguaje de ejemplo: Java

```
intentar {  
    Archivo archivo = nuevo Archivo ("objeto.obj");  
    ObjectInputStream en = nuevo ObjectInputStream(nuevo FileInputStream(archivo));  
    botón javax.swing.JButton = (javax.swing.JButton) in.readObject();  
    cercar();  
}
```

Lenguaje de ejemplo: Java

```
readObject final vacío privado (ObjectInputStream in) lanza java.io.IOException {  
    throw new java.io.IOException ("No se puede deserializar"); }  
}
```

Formas comunes en las que se puede explotar esta vulnerabilidad

Inyección de código malicioso: Un atacante puede modificar los datos serializados para incluir código malicioso que se ejecutará durante el proceso de deserialización.

Manipulación de datos: Un atacante puede manipular los datos serializados de manera que cause un comportamiento no deseado en la aplicación al deserializarlos.

Elevación de privilegios: Si la deserialización se realiza con privilegios elevados, un atacante podría lograr una elevación de privilegios al explotar la vulnerabilidad.

Para prevenir este tipo de vulnerabilidad, es importante seguir buenas prácticas de seguridad, como

Validación de entrada: Asegurarse de que los datos deserializados provengan de fuentes confiables y estén en el formato esperado.

Uso de mecanismos seguros: Utilizar bibliotecas y funciones que implementen mecanismos seguros de deserialización, como la verificación de la firma digital de los datos.

Principio de menor privilegio: Limitar los privilegios asignados al proceso de deserialización para minimizar el impacto en caso de una explotación.

Monitorización y registro: Implementar un sistema de registro robusto que registre intentos de deserialización no autorizados o sospechosos para facilitar la detección y respuesta a posibles ataques.



Conclusión

Para prevenir estas vulnerabilidades, es crucial seguir prácticas de seguridad, como validar la entrada, utilizar mecanismos seguros de deserialización, aplicar el principio de menor privilegio y establecer sistemas de monitorización y registro. La deserialización segura es una consideración crítica en el desarrollo de software seguro, y las organizaciones deben tomar medidas proactivas para mitigar el riesgo asociado con estas vulnerabilidades.