LA IMPORTANCIA DE LA PROGRAMACIÓN SEGURA EN EL DESARROLLO DE SOFTWARE

ALAN RODRIGO CORINI GUARACHI, Ph.D.

alan_corini@hotmail.com

Carrera de Informática

Universidad Mayor de San Andrés

La Paz, Bolivia

RESUMEN

La programación segura se ha convertido en un pilar fundamental para mitigar los riesgos cibernéticos en un entorno tecnológico cada vez más complejo.

A pesar de que las prácticas de desarrollo inseguro aún persisten, es necesario cambiar el paradigma hacia un enfoque que integre la seguridad como parte central del desarrollo de software, tomando como ejemplo el uso de consultas preparadas y la adopción de marcos de seguridad avanzados, como el modelo de Zero Trust (confianza cero), demuestran su efectividad en reducir vulnerabilidades críticas.

La implementación de metodologías como el SSDLC y SecDevOps, que incorporan controles de seguridad desde las fases iniciales de diseño, ha demostrado ser clave para disminuir las vulnerabilidades durante el ciclo de vida del software.

El informe de IBM Security (2024) destaca que las brechas de seguridad alcanzan costos sin precedentes, aunque también señala que el uso de tecnologías emergentes como la inteligencia artificial y la automatización está mejorando significativamente la detección y mitigación de riesgos. Si bien la seguridad absoluta es inalcanzable, el empleo de herramientas y estrategias adecuadas fortalece la resiliencia de los sistemas frente a las amenazas actuales.

1. INTRODUCCIÓN

En la actualidad, el software desempeña un papel fundamental en la mayoría de las actividades humanas. Desde aplicaciones que gestionan datos financieros hasta sistemas que controlan infraestructuras críticas, el software es la columna vertebral de una sociedad cada vez más digitalizada. Sin embargo, el incremento en el uso de estos sistemas ha traído consigo una creciente exposición a riesgos cibernéticos. Las brechas de seguridad y los ataques dirigidos a aplicaciones vulnerables se han convertido en un problema recurrente, evidenciando la necesidad de implementar prácticas de programación segura en el desarrollo de software.

La programación segura se define como el conjunto de metodologías, principios y herramientas orientadas a minimizar las vulnerabilidades en el software, desde su diseño hasta su implementación y mantenimiento. Este enfoque no debe considerarse un elemento adicional del desarrollo, sino una característica esencial del mismo. Estudios recientes han demostrado que integrar la seguridad desde las etapas iniciales del ciclo de vida del desarrollo de software reduce significativamente la aparición de errores y vulnerabilidades que, de otra forma, podrían ser explotados por atacantes malintencionados.

La magnitud de las amenazas actuales exige un enfoque proactivo. Según el informe de IBM Security (2023), más del 90 % de las brechas de seguridad reportadas en el último año estuvieron relacionadas con errores en el código o fallas en la configuración de sistemas. Este panorama pone de manifiesto que el costo de no implementar medidas de seguridad adecuadas es considerablemente mayor que el esfuerzo requerido para diseñar sistemas seguros desde el principio. Además, los ataques no solo afectan la integridad y la confidencialidad de los datos, sino que también tienen implicaciones económicas y reputacionales para las organizaciones involucradas.

2. MATERIALES Y MÉTODOS

Según los estudios de IBM El "Cost of a Data Breach Report 2024" de IBM revela que el costo promedio global de una violación de datos aumentó un 10% en comparación con el año anterior, alcanzando los 4,88 millones de dólares. Este incremento, el más significativo desde la pandemia, se atribuye principalmente a la interrupción de negocios y a los costos de respuesta posteriores a la brecha, como el soporte al cliente y la remediación. Además, el informe destaca que más de la mitad de las organizaciones afectadas están trasladando estos costos adicionales a sus clientes, lo que podría generar desafíos en mercados competitivos ya afectados por la inflación. El estudio también señala que la implementación de inteligencia artificial (IA) y automatización en la seguridad está resultando beneficiosa, reduciendo los costos de las brechas en aproximadamente 2,2 millones de dólares en algunos casos. Estas tecnologías están acelerando la identificación y contención de las violaciones, disminuyendo el tiempo de respuesta y mitigando el daño resultante. Sin embargo, persiste una escasez significativa de personal capacitado en ciberseguridad, con más de la mitad de las organizaciones afectadas enfrentando una grave falta de personal en este ámbito, una brecha que ha aumentado en dos dígitos respecto al año anterior. Esta carencia de profesionales capacitados se agrava con la adopción acelerada de la IA generativa en diversas funciones organizacionales, lo que introduce riesgos sin precedentes y ejerce una presión adicional sobre los equipos de ciberseguridad. (IBM, 2024)

Eset Latinoamérica recomienda no asumir la confiabilidad de ningún componente sin pruebas; establecer mecanismos de autenticación sólidos y difíciles de eludir; implementar procesos de autorización además de la autenticación; separar los datos de las instrucciones de control; y validar explícitamente todas las entradas al

sistema. Estas prácticas buscan minimizar riesgos y fortalecer la seguridad de las aplicaciones desde su concepción hasta su implementación final. (Giusto, 2015)

Existen varias metodologías de desarrollo de software como como (Micucci, 2023):

- **Desarrollo Seguro de Software (SSD):** Enfocado en identificar y mitigar vulnerabilidades desde el diseño hasta el mantenimiento.
- **Sec DevOps**: Combina prácticas de DevOps con seguridad, incorporando herramientas y prácticas de seguridad en el ciclo de desarrollo continuo.
- **Desarrollo Seguro por Diseño (SSDLC):** Integra controles de seguridad en los requisitos y la arquitectura del software desde la etapa de diseño.
- **Modelo de Madurez de Seguridad (SMM)**: Evalúa y mejora la madurez de la seguridad en el entorno de desarrollo, proporcionando un marco para identificar el estado actual y establecer metas de mejora.
- **Metodologías Ágiles y Seguridad (Agile Security)**: Combina metodologías ágiles como Scrum o Kanban con enfoques de seguridad, realizando revisiones continuas que incluyen análisis de seguridad de código y pruebas de penetración en cada iteración del desarrollo.
- · **Análisis de Amenazas y Modelado (TAM)**: Identifica amenazas y riesgos desde las etapas iniciales del desarrollo mediante técnicas como análisis de riesgos y modelado de amenazas.

3. RESULTADOS

La mayoría del código inseguro se debe a malas prácticas de seguridad además de no seguir las recomendaciones o metodologías de seguridad, por ejemplo se tiene los siguientes códigos seguros y como se debe implementar su implementación segura.

JAVA

Problema: El input se concatena directamente a la consulta SQL.

Código Vulnerable

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class SQLInjectionExample {

   public static void main(String[] args) {

        String userInput = "admin' OR '1'='1"; // Simula un input malicioso

        try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "user", "password");

        Statement stmt = conn.createStatement()) {
```

```
        String query = "SELECT * FROM users WHERE username = '" + userInput + "'";
        ResultSet rs = stmt.executeQuery(query);

        while (rs.next()) {
        System.out.println("User: " + rs.getString("username"));
        }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Solución: Se utiliza PreparedStatement, que evita la inyección SQL al manejar los parámetros de manera segura.

Código seguro:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
public class SecureSQLExample {
        public static void main(String[] args) {
        String userInput = "admin"; // Input del usuario
        try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "user", "password")) {
        String query = "SELECT * FROM users WHERE username = ?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setString(1, userInput); // Sustituir el parámetro con el input seguro
        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
```

```
        System.out.println("User: " + rs.getString("username"));
        }
        } catch (Exception e) {
        e.printStackTrace();
        }
        }
}
```

Python

Problema: El uso de f-strings o concatenación en las consultas permite inyecciones SQL.

Código Vulnerable

```
import sqlite3

def get_user_data(username):
        conn = sqlite3.connect('example.db')
        cursor = conn.cursor()

        # Input malicioso: 'admin' OR '1'='1'
        query = f"SELECT * FROM users WHERE username = '{username}'"
        cursor.execute(query)
        for row in cursor.fetchall():
        print(row)
        conn.close()
# Llamada a la función con input malicioso
get_user_data("admin' OR '1'='1")
```

Solución: Los placeholders (?) en la consulta y el uso de parámetros evitan la ejecución de código SQL no deseado.

Código seguro:

```
import sqlite3
def get_user_data(username):
        conn = sqlite3.connect('example.db')
        cursor = conn.cursor()
        # Consulta preparada con un placeholder (?)
        query = "SELECT * FROM users WHERE username = ?"
        cursor.execute(query, (username,))  # El parámetro se pasa como una tupla
        for row in cursor.fetchall():
        print(row)
        conn.close()
# Llamada a la función con un input seguro
get_user_data("admin")
```

PHP

Problema: El input del usuario se inserta directamente en la consulta SQL.

Código Vulnerable

```
<?php
// Conexión a la base de datos
$conn = new mysqli("localhost", "user", "password", "mydb");

if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
}
// Input del usuario (posiblemente malicioso)
$userInput = "admin' OR '1'='1";

// Consulta insegura
$query = "SELECT * FROM users WHERE username = '$userInput'";
```

```
$result = $conn->query($query);


if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
        echo "User: " . $row["username"] . "<br>";
        }
} else {
        echo "No results";
}


$conn->close();
?>
```

Solución: Se usa prepare y bind_param para parametrizar la consulta y evitar concatenaciones inseguras.

Código seguro:

```
<?php
// Conexión a la base de datos
$conn = new mysqli("localhost", "user", "password", "mydb");


if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
}
// Input del usuario
$userInput = "admin";
// Consulta preparada
$query = $conn->prepare("SELECT * FROM users WHERE username = ?");
$query->bind_param("s", $userInput); // "s" indica que el parámetro es un string
$query->execute();
```

```
$result = $query->get_result();
if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
        echo "User: " . $row["username"] . "<br>";
        }
} else {
        echo "No results";
}
$query->close();
$conn->close();
?>
```

4. DISCUSIÓN

La programación segura no es solo una medida preventiva, sino una estrategia esencial para mitigar riesgos cibernéticos y garantizar la integridad, confidencialidad y disponibilidad de los sistemas digitales, la implementación de prácticas seguras en el desarrollo de software tiene un impacto significativo en la reducción de vulnerabilidades.

A pesar de ello, persisten desafíos relacionados con la falta de concienciación y la ausencia de formación adecuada entre los desarrolladores, lo que subraya la importancia de fomentar una cultura organizacional centrada en la seguridad.

El desarrollo inseguro de software no solo esta aplicado a un lenguaje de programación, sino que todo lenguaje tiene sus ejemplos de cómo desarrollar de manera segura siguiendo diferentes metodologías de desarrollo seguro, en cada ejemplo donde se observó código inseguro también se planteó soluciones como el uso de consultas preparadas y parámetros seguros, demuestran que evitar vulnerabilidades como la inyección SQL es un proceso técnicamente factible y no necesariamente complejo.

Sin embargo, la dependencia de enfoques inseguros sigue siendo común debido a la presión por reducir tiempos de desarrollo o a la falta de conocimiento técnico en los equipos, por tal motivo se debe incluir la programación segura como parte integral en la formación en las universidades.

El informe de IBM Security (2024) también aporta una visión importante sobre el impacto económico de las brechas de seguridad, señalando que el costo promedio de una violación de datos ha alcanzado niveles históricos, por otro lado, el uso de tecnologías emergentes como la inteligencia artificial y la automatización ha mostrado ser una herramienta valiosa para identificar y mitigar riesgos de manera más eficiente.

Además de las prácticas técnicas, se debe destacar la importancia de los marcos de trabajo y metodologías, como el SecDevOps o el SSDLC, para integrar la seguridad en todas las etapas del desarrollo, el uso de estas

metodologías no solo fortalece la seguridad desde el diseño, sino que también promueven una mentalidad proactiva y colaborativa entre los equipos de desarrollo y seguridad.

El modelo de madurez de seguridad (SMM), por ejemplo, permite a las organizaciones identificar áreas de mejora y establecer metas realistas para incrementar su resiliencia frente a amenazas.

Aunque no existe un sistema completamente seguro, lo que se puede hacer es implementar medidas proactivas para reducir significativamente el riesgo de vulnerabilidades y ataques, la seguridad absoluta es un ideal inalcanzable debido a la constante evolución de las amenazas, las vulnerabilidades del día cero y a la complejidad de los sistemas modernos, que integran múltiples tecnologías, bibliotecas de terceros y componentes externos, aunque los desarrolladores y las organizaciones pueden minimizar el impacto de estas amenazas adoptando un enfoque de seguridad por capas, donde cada nivel del sistema cuenta con controles específicos diseñados para prevenir, detectar y mitigar posibles riesgos.

La reducción del riesgo implica una combinación de prácticas técnicas, como la validación de entradas, el uso de cifrado robusto y el monitoreo continuo, junto con estrategias organizacionales, como la capacitación regular de los equipos y la implementación de marcos como el modelo confianza cero, que si bien no es posible garantizar la seguridad total, sí es viable dificultar el acceso de los atacantes, reducir las superficies de ataque y reaccionar de manera oportuna ante posibles incidentes.

5. CONCLUSIONES

La programación segura se consolida como un elemento esencial en el desarrollo de software en un entorno donde las amenazas cibernéticas son cada vez más sofisticadas y frecuentes, las prácticas seguras desde las etapas iniciales del diseño, como el uso de consultas preparadas en la gestión de bases de datos y la adopción de metodologías como SecDevOps y SSDLC, no solo mitiga riesgos críticos como la inyección SQL, sino que también contribuye a mejorar la eficiencia operativa y a reducir los costos asociados a brechas de seguridad.

A pesar de los avances en tecnología y automatización, persisten desafíos como la falta de personal capacitado en ciberseguridad y la dependencia de prácticas de desarrollo obsoletas, para superar estas barreras, es imperativo fomentar la colaboración entre las instituciones educativas, las organizaciones y los desarrolladores, promoviendo una educación integral y el uso de estándares internacionales como OWASP, seguir las recomendaciones de empresas de Antivirus como ESET, Kaspersky Lab entre otros.

SOBRE EL AUTOR

Licenciado en Informática con mención en Ingeniería de Sistemas Informáticos, Doctor en Ciencias de la Computación y Magíster en Ciencias de la Computación con mención en Seguridad Informática y Software Libre. Cuenta con diplomado en Datos Abiertos y Contrataciones Abiertas.

Ha trabajado como Administrador de Base de Datos en Tycsys, desarrollador en el Ministerio de Economía y Finanzas Públicas y especialista en análisis y desarrollo de sistemas en el INE, utilizando tecnologías como Java, PHP, Angular y bases de datos Oracle, MySQL y PostgreSQL.

Su objetivo profesional es contribuir al desarrollo de sistemas informáticos eficientes y seguros, aplicando conocimientos en programación, desarrollo web, inteligencia artificial y máquinas de aprendizaje, así como liderar proyectos tecnológicos complejos.

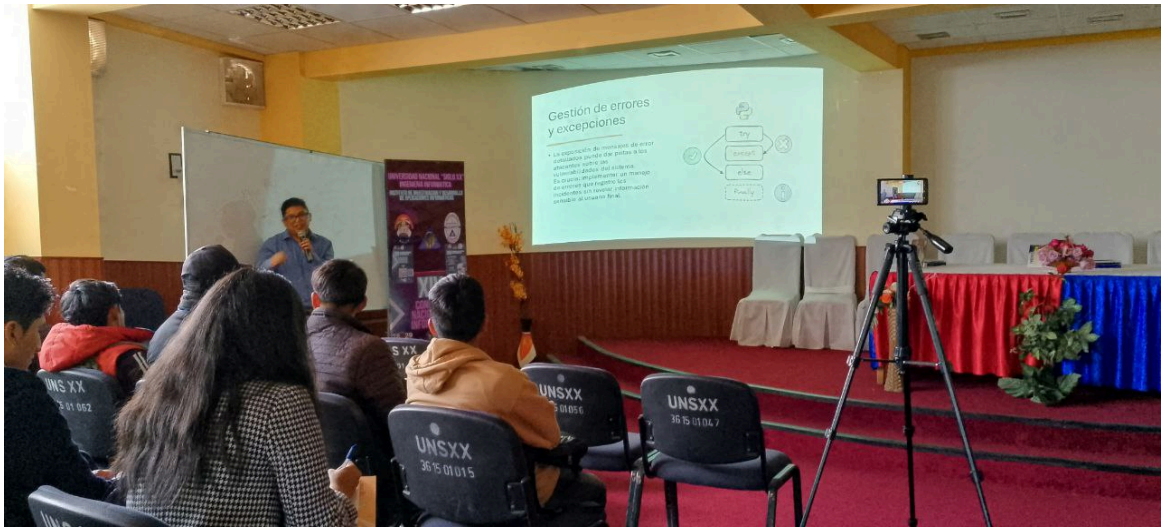

Figura 1: Fotografía de presentación de la ponencia