[English version]

Best Practices to Protect Applications from Cyber Threats

Introduction

Security in software development is essential, especially in an environment where threats evolve constantly. Even if you're not a cybersecurity expert, applying a few fundamental best practices can significantly reduce risks and strengthen your applications.

Background and Context

According to multiple security reports, more than **70% of successful attacks** exploit known but unpatched vulnerabilities. Common issues—such as insecure configurations or poor data validation—still account for a large number of incidents across the industry.

---

1. Implement Proper Data Validation and Sanitization

Most attacks like SQL injection or XSS occur due to unvalidated input.
Best practices:

- Validate input on both the client and server side.
- Use trusted sanitization libraries.
- Avoid manually building queries; rely on ORMs or parameterized queries.

Example:

A form that accepts raw user input without validation may allow an attacker to inject malicious scripts. Sanitization helps neutralize these payloads before they cause damage.

---

2. Keep Dependencies and Configurations Up to Date

Frameworks, packages, and servers require frequent updates.
Recommendations:

- Automate vulnerability monitoring.
- Avoid outdated or abandoned dependencies.

- Use secure default configurations whenever possible.

---

3. Implement Strong Authentication and Authorization

Access control mistakes can expose sensitive data.
Suggestions:

- Use modern standards like OAuth2 or JWT.
- Apply the principle of least privilege.
- Require MFA for critical access points.

Conclusion

Security doesn't have to be overwhelming. Validating input, keeping dependencies updated, and enforcing solid access controls can dramatically reduce vulnerabilities. Start by making these practices part of your development routine, and you'll be on a strong path toward building safer and more resilient applications.

[Versión en español]

Mejores prácticas para proteger aplicaciones de amenazas cibernéticas

Introducción

La seguridad en el desarrollo de software es un tema esencial, especialmente en un entorno donde las amenazas evolucionan constantemente. Aunque no seas especialista en ciberseguridad, aplicar buenas prácticas básicas puede reducir significativamente los riesgos y fortalecer tus aplicaciones.

Contexto y antecedentes

De acuerdo con distintos reportes de seguridad, más del **70% de los ataques exitosos** explotan vulnerabilidades conocidas y no corregidas. Además, errores comunes como configuraciones inseguras o validación deficiente de datos siguen siendo responsables de una gran cantidad de incidentes.

---

1. Implementa validación y sanitización de datos

La mayoría de ataques como inyección SQL o XSS ocurren por no validar entradas.
Buenas prácticas:

- Utiliza validaciones del lado del servidor y cliente.
- Emplea librerías de sanitización confiables.
- Evita construir consultas manuales; usa ORM o consultas parametrizadas.

Ejemplo:

Una app que recibe formularios sin validar podría permitir que un atacante envíe scripts maliciosos. Con sanitización, ese payload se neutraliza.

---

2. Mantén dependencias y configuraciones al día

Frameworks, paquetes y servidores requieren actualizaciones frecuentes.
Recomendaciones:

- Automatiza el monitoreo de vulnerabilidades.
- Evita dependencias obsoletas o abandonadas.

- Usa configuraciones seguras por defecto siempre que sea posible.

---

3. Implementa autenticación y autorización robusta

Errores en permisos pueden exponer datos sensibles.
Sugerencias:

- Usa estándares modernos como OAuth2 o JWT.
- Aplica el principio de mínimo privilegio.
- Requiere MFA para accesos críticos.

Conclusión

La seguridad no tiene que ser compleja. Validar entradas, mantener dependencias actualizadas y aplicar controles de acceso sólidos puede reducir de forma sustancial las vulnerabilidades. Inicia con estas prácticas y conviértelas en parte natural de tu ciclo de desarrollo para construir aplicaciones más seguras desde la base.