| **Risk ID** | **Vulnerability Name** | **Description** | **PetStore Application Example** | **Recommended Mitigation** |
| --- | --- | --- | --- | --- |
| **A01:2021** | Improper Access Control | The application allows unauthorized users to interact with data or features beyond their privilege level. | Users can view cart or order pages without logging in by directly entering the URL. | Enforce role-based access control both on the server and client sides. Use a "deny-by-default" approach and validate permissions on each request. |
| **A02:2021** | Weak Cryptographic Practices | Sensitive information is not protected during transmission or storage. | Passwords might be sent in plaintext and no mention of secure password handling. | Implement HTTPS on all pages and hash passwords with strong algorithms like bcrypt and salting. |
| **A03:2021** | Injection Attacks | Unsanitized user inputs are used in backend logic, enabling injection of malicious code. | The search input may allow SQL or script injection if input isn’t properly filtered. | Use prepared statements or parameterized queries. Apply strict input validation and output encoding. |
| **A04:2021** | Unsafe Design Practices | Lack of built-in security checks in application logic. | The cart allows users to enter negative or unrealistic quantities. | Use secure design principles such as input validation at every layer and misuse-case modeling during development. |
| **A05:2021** | Misconfigured Security Settings | Poorly configured system settings lead to security exposures. | Missing headers like HSTS or CSP; detailed server errors are shown to users. | Activate security headers, suppress detailed error outputs, and review server configurations regularly. |
| **A06:2021** | Outdated Software Components | Use of old or unpatched third-party software introduces vulnerabilities. | Application may rely on deprecated JavaScript libraries. | Maintain an inventory of dependencies and regularly scan with tools like npm audit. Keep all components updated. |
| **A07:2021** | Authentication and Session Flaws | Weaknesses in login mechanisms and session controls. | Login page lacks CAPTCHA; session remains active after logout; inadequate password policies. | Enforce robust password requirements, include CAPTCHA, implement session timeouts, and secure session tokens. |
| **A08:2021** | Lack of Data Integrity Assurance | Absence of integrity checks for software updates or third-party code. | No Subresource Integrity (SRI) for external scripts; deployment pipeline lacks verification. | Use SRI for third-party content, implement secure CI/CD practices, and ensure all components are verified. |
| **A09:2021** | Inadequate Logging and Monitoring | The application does not log key events or detect malicious behavior. | Failed login attempts and unusual transactions go unmonitored. | Enable centralized logging, monitor suspicious activities, and configure real-time alerts for anomalies. |
| **A10:2021** | Server-Side Request Forgery (SSRF) | The server processes attacker-controlled requests to internal services. | If URLs are accepted (e.g., for profile pictures), attackers could access internal systems. | Filter URLs, enforce domain whitelisting, and prevent internal network access through proper validation. |