SECURITY DEVELOPMENT PRACTICES QUESTIONNAIRE

Based on the Microsoft SDL.

Training

1. **Do all stakeholders follow security training?**
   Security training should cover foundational concepts for building better software include secure design, threat modeling, secure coding, security testing, and best data surrounding privacy.

Requirements

1. **Are security and privacy requirements defined early?**
   Having security and privacy requirements defined early makes it easier to identify key milestones and deliverables and minimize disruptions to plans and schedules.

2. **Are minimum acceptable levels of security and privacy quality defined at the start?**
   Helps a team understand risks associated with security issues, identify and fix security bugs during development, and apply the standards throughout the entire project.

3. **Is software design examined based on costs and regulatory requirements?**
   Helps a team identify which portions of a project will require threat modeling and security design reviews before release and determine the Privacy Impact Rating of a feature, product, or service.

Design

1. **Are security design requirements defined?**
   Considering security and privacy concerns early helps minimize the risk of schedule disruptions and reduce a project's expense.

2. **Is the attack surface analysis and pro-actively reduced?**
   Reducing the opportunities for attackers to exploit a potential weak spot or vulnerability requires thoroughly analyzing overall attack surface and includes disabling or restricting access to system services, applying the principle of least privilege, and employing layered defenses wherever possible.

3. **Is threat modeling is performed?**
   Applying a structured approach to threat scenarios during design helps a team more effectively and less expensively identify security vulnerabilities, determine risks from those threats, and establish appropriate mitigations.

Implementation

1. **Are only approved tools used?**
   Publishing a list of approved tools and associated security checks (such as compiler/linker options and warnings) helps automate and enforce security data easily at a low cost. Keeping the list regularly updated means the latest tool versions are used and allows inclusion of new security analysis functionality and protections.

2. **Are unsafe functions banned? How?**
   Analyzing all project functions and APIs and banning those determined to be unsafe helps reduce potential security bugs with very little engineering cost. Specific actions include using header files, newer compilers, or code scanning tools to check code for functions on the banned list, and then replacing them with safer alternatives.

3. **Is static code analysis performed?**
   Analyzing the source code prior to compile provides a scalable method of security code review and helps ensure that secure coding policies are being followed.

Verification

1. **Is dynamic analysis performed?**
   Performing run-time verification checks software functionality using tools that monitor application behavior for memory corruption, user privilege issues, and other critical security problems.

2. **Is fuzz testing performed?**
   Inducing program failure by deliberately introducing malformed or random data to an application helps reveal potential security issues prior to release while requiring modest resource investment.

3. **Is attack surface reviewed?**
   Reviewing attack surface measurement upon code completion helps ensure that any design or implementation changes to an application or system have been taken into account, and that any new attack vectors created as a result of the changes have been reviewed and mitigated including threat models.

Release

1. **Is incident response plan created?**
   Preparing an Incident Response Plan is crucial for helping to address new threats that can emerge over time. It includes identifying appropriate security emergency contacts and establishing security servicing plans for code inherited from other groups within the organization and for licensed third-party code.

2. **Are final security reviews performed?**
   Deliberately reviewing all security activities that were performed helps ensure software release readiness. The Final Security Review (FSR) usually includes examining threat models, tools outputs, and performance against the quality gates and bug bars defined during the SecurityRequirements Phase.

3. **Are release and archive certified?**
   Certifying software prior to a release helps ensure security and privacy requirements were met. Archiving all pertinent data is essential for performing post-release servicing tasks and helps lower the long-term costs associated with sustained software engineering.

Response

1. **Is incident response plan executed? How?**
   Being able to implement the Incident Response Plan instituted in the Release phase is essential to helping protect customers from software security or privacy vulnerabilities that emerge.