**Security Engineering**

**Secure Coding Guidelines:**

Es gibt 2 Klassen von Guidelines: allgemeine (general) oder sprachabhängige (language specific)

**General Guidelines:**

* white lists sind besser als black lists zur Validierung der Benutzereingaben
* KISS-Principle
* Versions- und Konfigurationskontrolle
* Benutzung von Security Design Patterns
* keine Nutzung von unsicheren Funktionen (z.B. gets())
* ...

**Security Patterns:**

**Struktur:**

* Kontext

- Beispielszenario, Annahmen über das IT-System, ...

* Problem

- Gefahren, gegen die man sich schützen möchte

* Anforderungen
* Lösungen

- verschiedene Lösungen werden vorgestellt (Diskussion Vor- und Nachteile)

- Ziel: Risikominimierung

* Verwandte Muster

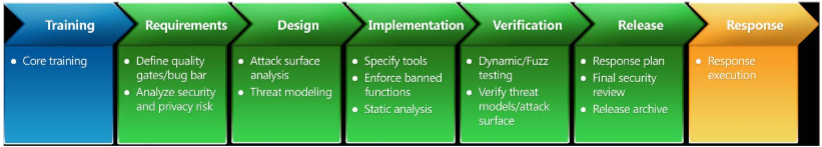
**Klassen:**

* Architectural-level patterns
* Design-level patterns
* Implementation-level patterns

**Privilege Seperation:**

* Code in Teile, die limitierte Privilegien benötigen und Teile, die spezielle Privilegien benötigen, aufgeteilt
* nur die Teile, die spezielle Privilegien benötigen, werden mit diesen Privilegien auf dem Server ausgeführt, der Rest mit den limitierten

**SDL:**

****

1. Training
2. Requirements

* Spezifikation der Sicherheits- und Privatsphäreanforderungen
* Defining Quality Gates: Qualitätskriterien, die über die Freigabe des nächsten Projektschritts entscheiden
* Security and Privacy Risk Assessment: Identifizieren funktionale Aspekte, die eine genauere Überprüfung erfordern

1. Desing

* Definiert und dokumentiert Sicherheitsarchitektur

1. Implementation

* Durchsetzung von Sicherheitspraktiken, um sichere Softwareentwicklung sicherzustellen

1. Verification

* Es wird getestet, ob die Software die spezifizierten Sicherheits- und Privatsphäreanforderungen erfüllt

1. Release

* Software wird auf Auslieferung vorbereitet

1. Response

Buffer-Overflows noch in Bearbeitung