

# Wesentliche Designmerkmale Sicherer Software

EUROSEC GmbH Chiffriertechnik & Sicherheit

Autor: Dr. Thilo Zieschang, EUROSEC

Tel: 06173 / 60850, [www.eurosec.com](http://www.eurosec.com)

# Vortragsübersicht

- Sicherheitsziele und Anforderungsanalyse
- Was versteht man unter Security Design?
- Welche Designkriterien sind unverzichtbar für die Produktsicherheit?



# Sicherheitsziele und Anforderungsanalyse

# Sicherheitsziele für Softwareprodukte

- zunächst die klassischen, generischen Sicherheitsziele bzw. –eigenschaften:
  - (Wahrung der) Vertraulichkeit
  - (Wahrung der) Authentizität und Integrität
  - (Wahrung der) Verfügbarkeit
- zusätzlich jedoch, quasi auf Metaebene:
  - Nachweisbarkeit des Erreichens o.g. Ziele durch die Entwickler
  - Unterstützung der Erreichbarkeit o.g. Ziele durch den späteren Betreiber
  - Angemessenheit hinsichtlich potentieller Einsatzzwecke

# Anforderungsanalyse

- Anforderungen identifizierter Einsatzszenarien analysieren
- individ. Risikobewertung und/oder Baseline Security?
- Anforderungen externer Rahmenwerke? (z.B. COBIT, ITIL, ISO17799, BSI Grundschutz, Sarbanes Oxley, FDA, Bankenaufsicht, Datenschutzgesetz, NIST, sonstige Industriestandards oder Rahmenwerke?)
- spezifische Wünsche von Kunde bzw. Auftraggeber?

# Anforderungsspezifikation

- alle identifizierten Anforderungen müssen präzise formuliert und dokumentiert werden
- mehrere Iterationen zweckmäßig, erst allgemein, dann zunehmend detailliert
- zu große Interpretationsspielräume vermeiden
- für jede Anforderung muss später klar entscheidbar sein, ob diese erfüllt wurde oder nicht
- vorgegebene Gliederung und Syntax empfohlen, im Einzelfall auch Verwendung spezieller Tools

# Identifizierung verbleibender Restrisiken

- aus wirtschaftlichen und/oder technischen Gründen können nie alle wünschenswerten Sicherheitsanforderungen erzwungen werden
- daher sind verbleibende Restrisiken bewusst zu identifizieren und zu dokumentieren
- Für alle Restrisiken sollte entschieden werden:
  - Entschärfung durch externe technische Lösung?
  - Entschärfung durch organisatorische Maßnahmen?
  - keine Gegenmaßnahme, ABER: Quantifizierung und Worst Case Betrachtung und bewusste Inkaufnahme der möglichen Schadensauswirkungen?
  - andere Lösung, beispielsweise Versicherung o.ä.?



# Was versteht man unter Security Design?

# Abgrenzungsversuche

- in der Theorie differenziert man
  - Funktionale Anforderungen
  - Architekturansforderungen
  - Designanforderungen
  - Implementierungsanforderungen
  - Prozessanforderungen
  - Prüfanforderungen
  - etc. etc.
- in der Praxis findet sich keine allgemein gültige Abgrenzung, manche Aspekte sind schwer einzuordnen

# Funktionale Anforderungen

- Welche konkrete Funktionalität soll die Software bieten?
- Beispielanforderung: „Administrierbarkeit muss in vollem Funktionsumfang auch ohne grafische Benutzeroberfläche mittels Kommandozeile möglich sein. Parameter sollen in Konfigurationsdateien ein- und auslesbar sein.“
- aber: dies könnte auch in eine Designanforderung übersetzt werden (müssen)

# Architektur Anforderungen

- oder Bezeichnung als : „High Level Designkriterien“
  - z.B. Positionierung im Netzwerk, Aufteilbarkeit auf verschiedene Maschinen und Segmente, Trennung elementarer funktionaler Schichten (horizontal und vertikal), etc.
  - Beispielanforderung: „Benutzerdaten werden in einer gesonderten Datenbank außerhalb der Applikation gehalten“
- Aber auch Überschneidungen, die ebenso Designmerkmale wären, beispielsweise:
- „Verteilbarkeit der Applikation auf mehrere Maschinen nebst Hot Swap Fähigkeit bei Ausfall einer Maschine.“

# Designanforderungen

- Wesentliche Gestaltungsmerkmale, häufig nicht funktionaler Natur, und großteils unabhängig von Implementierungsdetails
- Beispielanforderung: „Anwendung soll im Fehlerfall möglichst in einen „sicheren“ Zustand übergehen.“

# Implementierungsanforderungen

- Während der Implementierung zu berücksichtigen, in der Regel abhängig von Programmiersprache, Plattform, Entwicklungsumgebung etc.
- Beispielanforderung: „Schutz gegen Buffer Overflows.“

Aber auch Überschneidungen, die ebenso Designmerkmale wären, beispielsweise:

- „Keine Passworte im Klartext abspeichern.“,  
„Keine Passworte auf dem Bildschirm anzeigen.“

# Prozessanforderungen

- Anforderungen zur Unterstützung bzw. Umsetzbarkeit technisch/organisatorischer Begleitprozesse der Produktsicherheit
- Beispielanforderung: „Einrichtung eines Security Bug Tracking und Reporting Tools.“

Aber auch Überschneidungen, die ebenso Designmerkmale wären, beispielsweise:

- „Bereitstellung geeigneter Patch- und Updatemechanismen.“

# Prüfanforderungen

- Überprüfbarkeitskriterien hinsichtlich Einhaltung formulierter Sicherheitsvorgaben
- Beispielanforderung: “Es ist ein Source Code Checker bereitzustellen und anzuwenden, der auf Verwendung verbotener Sprachkonstrukte prüft.“

Aber auch Überschneidungen, die ebenso Designmerkmale wären, beispielsweise:

- „Nichtautorisierte Änderungen der installierten Code Basis müssen festgestellt werden können.“

# „Designziele“ oder „Sicherheitsziele“?

- Die folgenden Begriffe sind eigentlich keine Designziele, sondern Sicherheitsziele:
  - Identifizierung und Authentisierung
  - Zugriffskontrolle und Autorisierung
  - Beweissicherung und Nachvollziehbarkeit
  - Übertragungssicherung
  - Datenablagensicherung
  - Robustheit
  - Verfügbarkeit

Welche Designkriterien sind  
unverzichtbar für die  
Produktsicherheit?

# Einfachheit anstreben

- höhere Komplexität bedeutet höhere Fehleranfälligkeit
- überflüssige Funktionalität (auch hinsichtlich Security) vermeiden
- meist existieren unterschiedlich aufwändige Alternativen zur Erreichung einer gegebenen Funktionalität
- auch Folgekosten (Wartung, Bug Fixing) berücksichtigen...

# Anwendbarkeit / Usability

- „Keep it simple“ gilt nicht nur für Innenleben, sondern auch für Außensicht
- Intuitive Bedienbarkeit anstreben
- keine sicherheitsrelevanten Einstellungen in Untermenüs verstecken
- Praxistests mit Testanwendern hinsichtlich Praktikabilität
- wo immer möglich, sollte sich Konfiguration auch ohne Handbuch erschließen

# Höchstmögliche Akzeptanz anstreben

- selbsterklärende und fehlerfrei bedienbare Sicherheitsfunktionen implizieren nicht notwendig Anwenderakzeptanz
- jeder erzwungene Mehraufwand reduziert die Akzeptanz
- bei gleichwertiger Funktionalität stets der weniger umständlichen Lösungsvariante den Vorzug geben
- Akzeptanztests durchführen und Feedback berücksichtigen

# Wirtschaftlichsten Mechanismus anwenden

- Akzeptanz von Sicherheitsmaßnahmen muss auch nach Innen vorhanden sein (z.B. Entwicklungsleiter, Produktmanager o.ä.)
- eine Lösung, die geringfügig weniger sicher ist aber nur einen Bruchteil Kosten verursacht, ist oftmals zu bevorzugen
- verfügbares Gesamtbudget für Security sollte mit Bedacht aufgeteilt werden, auch hinsichtlich Sicherheitsfunktionalität

# Vollständige Offenlegung der Funktionsweise voraussetzen

- Verfahrenssicherheit sollte nie darauf angewiesen sein, dass Funktionsdetails geheimgehalten werden
- unter dieser Prämisse sollte die Software dennoch sicher sein
- impliziert nicht notwendig, dass Details und/oder Source Code tatsächlich offengelegt werden
- Bewertung der Sicherheit setzt jedoch hypothetisch voraus, dass alle Details bekannt sind, einschließlich Source Code

# Sparsam mit Vertrauensbeziehungen agieren

- möglichst wenige Vertrauensbeziehungen einräumen, zum Beispiel bei:
  - Rehtedelegation an andere Programme, Benutzer oder Maschinen
  - Einräumung von Zugriffsrechten für authentifizierte Benutzer auf fremden Maschinen
  - Verzicht nochmaliger Berechtigungsprüfung bei sicherheitskritischen Aktionen
  - Zugriffsrechte administrativer Benutzer auf Ressourcen Dritter
  - Einrichtung von Zugangsberechtigungen über Netzsegment-Grenzen hinweg

# Protokollierung ermöglichen

- ausreichende Informationen protokollieren, um Tatverlauf ggf. zu rekonstruieren
- nicht nur Applikations-Schicht betrachten, sondern auch Betriebssystem und angrenzende Datenbanken u.ä. berücksichtigen (schlüssiges Gesamtbild anstreben)
- Schutz gegen Manipulation
- ggf. auch Schutz vor Lesezugriff durch beliebige Administratoren

# Fehlererkennung und – behebbarkeit anstreben

- Hilfestellungen vorsehen, die im Fehlerfall die Ursachenfindung und Behebung erleichtern
- hierbei darauf achten, dass mögliche Angreifer keine Angriffs-unterstützenden Details erfahren

# Sicheren Zustand im Fehlerfall gewährleisten

- Fehler können nie vollständig verhindert werden
- im Fehlerfall sollte jedoch stets ein möglichst sicherer Zustand erreicht werden
- Beispiel: abgestürzte Firewall schaltet nicht auf „Durchzug“, sondern auf Blockade
- zweckmäßiges Fehlerverhalten bei einem Röntgengerät ist voraussichtlich anders als bei einem Beatmungsgerät...

# Datenintegrität sicherstellen

- leichter gesagt als getan...
- Ziel: Unverfälschtheit abgespeicherter oder übertragener Daten und Programme
- Für Fehlererkennung und –korrektur sind zahlreiche Verfahren dokumentiert
- bei absichtlicher Manipulation muss jedoch auch mit manipuliertem Erkennungsmechanismus gerechnet werden; Schutz nur begrenzt praktikabel

# Standardmäßige Abweisung praktizieren

- alle Aktionen, die nicht explizit erlaubt wurden, sollen standardmäßig verboten sein

# Patch- bzw. Updatefähigkeit unterstützen

- Mechanismen müssen möglichst komfortabel sein, automatisierbar, und für große Anzahl Systeme skalierbar
- individuell vorgenommene Sicherheitseinstellungen dürfen durch Patch nicht verloren gehen oder heruntergesetzt werden

# Geringst mögliche Privilegien zugestehen

- jedem Benutzer und jedem Programm sollten nur jene Rechte eingeräumt werden, die diese für die Ausführung ihrer Aufgaben wirklich brauchen

# Komplette Rechteprüfung durchführen

- Jeder Zugriff auf ein Objekt sollte individuell geprüft werden
- nicht nur beim ersten Zugriff mit anschließendem Caching der Berechtigungsprüfung
- kein „Verstecken“ von Rechten, Funktionen oder Daten
- keine „Visibility Control“ (z.B. URLs, die nicht als Link angeboten werden, aber bei manueller Eingabe ungeprüft die zugehörige Seite ausgeben)

# Geringstmögliche gemeinsame Mechanismen und Ressourcen anstreben

- Separierung gemeinsamer Speicherbereiche anstreben
- alle gemeinsam genutzten Ressourcen stellen potentielle Problemstellen dar

# Verfügbarkeitsanforderungen im Design berücksichtigen

- Verfügbarkeitsanforderungen müssen im Design berücksichtigt und technisch unterstützt werden
- Beispiele:
  - Patches sollten bei Problemen zurückgerollt werden können
  - Verteilbarkeit auf mehrere Maschinen unterstützen
  - Mechanismus zum Klonen der Anwendung bereitstellen
  - effizientes Backup und Recovery unterstützen
  - automatisches Wiederhochfahren abgestürzter Prozesse
  - Bedienfehler mit kritischen Auswirkungen bestmöglich unterbinden

# Zugangsdaten und Verschlüsselungsschlüssel möglichst sicher ablegen

- keine sensiblen Schlüssel oder Passworte im Klartext ablegen
- im System abgelegte Schlüssel bestmöglich verstecken und verschleiern
- Betriebssystem-spezifische Schutzmechanismen hierfür nutzen (insbes. in Windows)

# Anbindung von Datenbanken und Backendsystemen sicher gestalten

- Vermeidung eines einzelnen, hochprivilegierten technischen Benutzers
- keine unsichere Ablage des Datenbankpasswortes auf dem Applikationsserver
- differenzierte Verwaltung von Zugriffsrechten und –berechtigten (auch) auf Datenbankseite
- Vermeidung doppelter Benutzerverwaltung auf Applikation einerseits und Datenbank andererseits
- Ablage von Benutzerberechtigungsprofilen bevorzugt in der Datenbank statt in der Applikation

# Hinreichend starke Sicherheitsfunktionen und –mechanismen wählen

- Auswahl der „richtigen“ Sicherheitsfunktionen und –mechanismen sollte nicht im Belieben der Entwickler liegen, sondern in der Designphase entschieden werden, ggf. gemeinsam mit Entwicklern
- ansonsten Gefahr schwacher, ungeeigneter oder schwer implementierbarer Mechanismen
- Standardbeispiel: schwache Verschlüsselung von Benutzerdaten, löchrige Datenfilter, schwache Lizenzschutzmechanismen, etc. etc.

# Warum dieser Vortrag von uns?

## - Unsere Erfahrung:

- mehrere Personenjahre in Forschungsprojekten zur sicheren Softwareentwicklung; derzeit gemeinsam mit Partnern wie SAP, Commerzbank, Universitäten, ...
- zahlreiche Schwachstellenanalysen für Softwarehersteller, nebst intensiver Feedbackzyklen mit den Entwicklern
- Erstellung von Anforderungs- und Designspezifikationen in mehreren großen Entwicklungsprojekten
- Erstellung von Guidelines zur sicheren Softwareentwicklung, mit Schwerpunkten Banking & Finance, sowie Webapplikationen
- Reverse Engineering und Gutachten von Sicherheitsfunktionen und Kryptomechanismen
- Implementierung von Sicherheitsfunktionen im Auftrag

# Abschlussbemerkung

- die vorliegende Dokumentation wurde von EUROSEC erstellt im Rahmen des secologic Forschungsprojekts, Laufzeit 2005 und 2006, nähere Informationen unter [www.secologic.org](http://www.secologic.org)
- wir bedanken uns beim Bundesministerium für Wirtschaft für die Förderung dieses Projektes
- Anregungen und Feedback sind jederzeit willkommen, ebenso Anfragen zu Sicherheitsaspekten, die hier nicht behandelt werden konnten.

# Copyright Hinweis

- Diese Folien wurden von EUROSEC erstellt und dienen der Durchführung von Schulungen oder Seminaren zum Thema Sichere Anwendungsentwicklung, mit Fokus Webapplikationen.
- Wir haben diese Folien veröffentlicht, um die Entwicklung besserer Softwareprodukte zu unterstützen.
- Die Folien dürfen gerne von Ihnen für eigene Zwecke im eigenen Unternehmen verwendet werden, unter Beibehaltung eines Herkunfts-Hinweises auf EUROSEC.
- Eine kommerzielle Verwertung, insbesondere durch Schulungs- oder Beratungsunternehmen, wie beispielsweise Verkauf an Dritte oder ähnliches ist jedoch nicht gestattet.