

## Secologic

# „Die Zehn Goldenen Regeln“ der IT-Sicherheit

Commerzbank / SAP

Version 1.0 – Dezember 2006

Dieses Dokument wurde von der SAP AG und der Commerzbank AG als Informationsmaterial erstellt und gibt keine Garantie für Vollständigkeit und Korrektheit jeglicher Art. Diese Arbeit wurde im Rahmen des vom BMWi (Bundesministerium für Wirtschaft und Technologie) öffentlich geförderten Projektes „*Secologic*“ erstellt. Wir danken dem BMWi für die Förderung dieses Projektes. Weitere Informationen finden Sie auf unserer Webseite [www.secologic.de](http://www.secologic.de).

## Inhaltsverzeichnis:

Einleitung .....	6
Zielgruppen.....	6
Überblick der 10 Goldenen Regeln für die beteiligten, unterschiedlichen Rollen im Softwarelebenszyklus .....	7
<b>1 DIE 10 GOLDENEN SICHERHEITS-REGELN FÜR „AUFTRAGGEBER“ (FACHABTEILUNGEN/ANALYSTEN) .....</b>	<b>8</b>
1.1 Denken Sie über geschäftliche Konsequenzen von IT-Sicherheitsfragen nach (Think about Business Impact of Potential Security Breaches) .....	8
1.2 Berücksichtigen Sie gesetzliche und regulatorische Anforderungen (Check Legal and Regulatory Requirements) .....	9
1.3 Definieren Sie fachliche Rollen, Zuständigkeiten und Prozesse (Define business roles, responsibilities and processes) .....	9
1.4 Berücksichtigen Sie auch Anforderungen an die Nachvollziehbarkeit (Consider also “Traceability” Requirements) .....	10
1.5 Verfügbarkeit und Business Continuity können spezielle Anforderungen sein (Availability and Business Continuity can be special Requirements) .....	10
1.6 Sicherheitsanforderungen sollten bekannt, vereinbart und fixiert sein (Security Requirements should be known, agreed and signed off) .....	11
1.7 Die Erfüllung von Sicherheitsanforderungen sollte abnahmerelevant sein (Fullfilment of Security requirements should be subject of acceptance) .....	11
1.8 Beachten Sie Restrisiken und treffen Sie ggf. Vorkehrungen (Be aware of remaining security risks and be prepared) .....	12
1.9 Vergessen Sie nicht Ihre eigenen Prozesse, organisatorischen Maßnahmen und Leute (Don’t forget your own Processes, Organizational Measures and People) .....	13
1.10 Alles kann sich ändern, seien Sie darauf vorbereitet (Everything can be subject to change, be prepared) .....	13
<b>2 DIE 10 GOLDENEN SICHERHEITS-REGELN FÜR „PLANER“ (ARCHITEKTEN) .....</b>	<b>14</b>
2.1 Nutzen Sie die Erfahrung von anderen .....	14
2.1.1 Erfinden Sie Schwachstellen nicht neu .....	14

2.1.2 Entwickeln Sie keine eigenen Sicherheitsfunktionen!..... 14

2.1.3 Internationale Sicherheitsstandards (technische Standards) ..... 14

2.2 „Minimale Rechte“-Prinzip, Berechtigungskonzept erstellen ..... 14

2.3 Trust Zones: Welchen eingehenden Daten darf vertraut werden und welchen nicht..... 15

2.4 Machen Sie es einfach! ..... 15

2.5 Niemals „Security by Obscurity“ ..... 15

2.6 Sichere Voreinstellungen..... 15

2.7 Machen Sie ein Architekturreview..... 16

2.8 Mehrere Sicherheitshürden (Defense in Depth)..... 16

2.9 Rechtliche Anforderungen beachten ..... 16

    2.9.1 Datenschutz ermöglichen..... 16

    2.9.2 Machen Sie alles nachvollziehbar (sofern sinnvoll)!..... 16

2.10 Rechnen Sie mit Schwachstellen: sehen Sie Patches vor ..... 17

**3 DIE 10 GOLDENEN SICHERHEITS-REGELN FÜR „DEVELOPER“ (ENTWICKLER)..... 18**

3.1 Misstrauen Sie jeder (Benutzer-) Eingabe (Validate Input) ..... 18

3.2 „Minimale Rechte“-Prinzip ..... 18

3.3 Keine unnötige Veröffentlichung von internen Daten! (Sanitize Output) ..... 19

3.4 Nutzen Sie die Erfahrung von anderen (do’s und dont’s)..... 19

    3.4.1 Erfinden Sie Schwachstellen nicht neu ..... 19

    3.4.2 Entwickeln Sie keine eigenen Sicherheitsfunktionen!..... 19

3.5 Sichere Einstellungen ..... 20

    3.5.1 Sichere Voreinstellungen ..... 20

    3.5.2 Alle Benutzerkennungen, Passwörter und Verschlüsselungsschlüssel änderbar!..... 20

    3.5.3 Machen Sie es einfach!..... 20

    3.5.4 Sicherheitsdokumentation..... 20

3.6 Niemals „Security by Obscurity“ und niemals „Hintertüren“ ..... 20

    3.6.1 „Security by Obscurity“ ..... 20

    3.6.2 Keine „Hintertüren“ ..... 21

3.7 Gute Fehlerbehandlung (Fail securely) ..... 21

3.8 Denken Sie an Nachvollziehbarkeit (Trace, Audit, Logs) ..... 21

3.9 Machen Sie Code-Reviews ..... 21

3.10 Rechnen Sie mit Schwachstellen: sehen Sie Patches vor ..... 22

**4 DIE 10 GOLDENEN SICHERHEITS-REGELN FÜR „TESTER“ ..... 23**

4.1 Testen Sie priorisiert Hauptangriffspunkte ..... 23

4.2 Machen Sie Code-Reviews ..... 23

4.3 Machen Sie Regressionstests ..... 23

4.4 Binden Sie externe Sicherheitsexperten mit ein (für Schwachstellentest und/oder – reviews) ..... 23

4.5 Nutzen Sie vorhandene Testfälle und Tools ..... 23

4.6 Testen Sie funktionale Sicherheitsanforderungen ..... 24

4.7 Testen Sie auf Schwachstellen ..... 24

4.8 Erstellen Sie einen Testplan..... 24

4.9 Führen Sie Sicherheitstests nicht gleichzeitig mit fachlichen Tests durch..... 24

4.10 Sehen Sie für alle ("normalen") Testfälle Negativbeispiele vor..... 24

**5 DIE 10 GOLDENEN SICHERHEITS-REGELN FÜR „IT-PROJEKTLLEITER“ ..... 25**

5.1 Sie sind verantwortlich für die Umsetzung aller Anforderungen, lesen Sie zunächst also alle anderen "Goldenen Regeln" (You are responsible for the implementation of all requirements, first read all other „Golden Rules“) ..... 25

5.2 Fordern Sie die fachlichen Sicherheitsanforderungen ein (Force the Business to clarify the Security Requirements) ..... 25

5.3 IT-Sicherheit gleich von Beginn an, nicht erst am Ende oder hinterher (IT-Security already at the beginning, not only at the end or afterwards)..... 26

5.4 In allen Projektphasen gibt es Arbeitspakete für IT-Sicherheit (Consider IT-Security in all project phases)..... 26

5.5 Planen Sie hinreichende Ressourcen für IT-Sicherheit (Plan sufficient resources for IT-Security) ..... 27

5.6 Vergessen Sie nicht Ihre (Projekt-)Mitarbeiter, planen Sie Sensibilisierung und Qualifizierung (Dont forget your employees, plan awareness and knowledge) ..... 27

5.7 Nutzen Sie Standards, sichere Komponenten und existierende Erfahrungen (Use and Reuse Standards, Trusted Components and existing Experience) ..... 27

5.8 Bewahren Sie Handlungsspielraum für mögliche Veränderungen (Have Options for possible changes) ..... 28

5.9 Machen Sie Restrisiken transparent und lassen sich diese abzeichnen (Achieve Awareness about Residual Risks and let them „Sign Off“) ..... 29

5.10 Denken Sie an diejenigen, die später mit dem Produkt umgehen müssen (Consider those people, who later have to work with the results) ..... 29

## Einleitung

Jeder Softwarelebenszyklus beinhaltet – egal nach welchem Vorgehensmodell vorgegangen wird – die verschiedenen, allgemeingültigen Prozessphasen *Analyse, Design, Entwicklung, Test* und *Integration*. Die anschließende *Wartung im Betrieb* gehört nicht mehr direkt zum Entwicklungsprozess aber zum Lebenszyklus der Software. Spricht man in der Öffentlichkeit von ‚Sicherheit‘ im Kontext mit Software wird von vielen die Verantwortung für Sicherheit beim Administrator im laufenden Betrieb gesehen. Dass IT-Sicherheit aber schon von Anfang an und über die gesamte Laufzeit des Softwareentwicklungszyklus bedacht werden sollte ist eine noch nicht sehr verbreitete Tatsache. Wie auch bei funktionalen Anforderungen bewahrheitet sich für die Anforderungen an die IT-Sicherheit, je früher und konsequenter Sicherheitsanforderungen an die zu erstellende Software gestellt werden, um so kostengünstiger wird die IT-Sicherheit im Produkt.

Jeder Phase im Softwarelebenszyklus lässt sich eine Rolle zuordnen, wobei gerade in mittelständischen Betrieben häufig von einer Person mehrere Rollen gleichzeitig eingenommen werden.

In der folgenden Abbildung haben wir eine Zuordnung der Phasen im Softwarelebenszyklus (obere Pfeilreihe) zu den einzelnen Rollen (beiden unteren Pfeilreihen) vorgenommen.

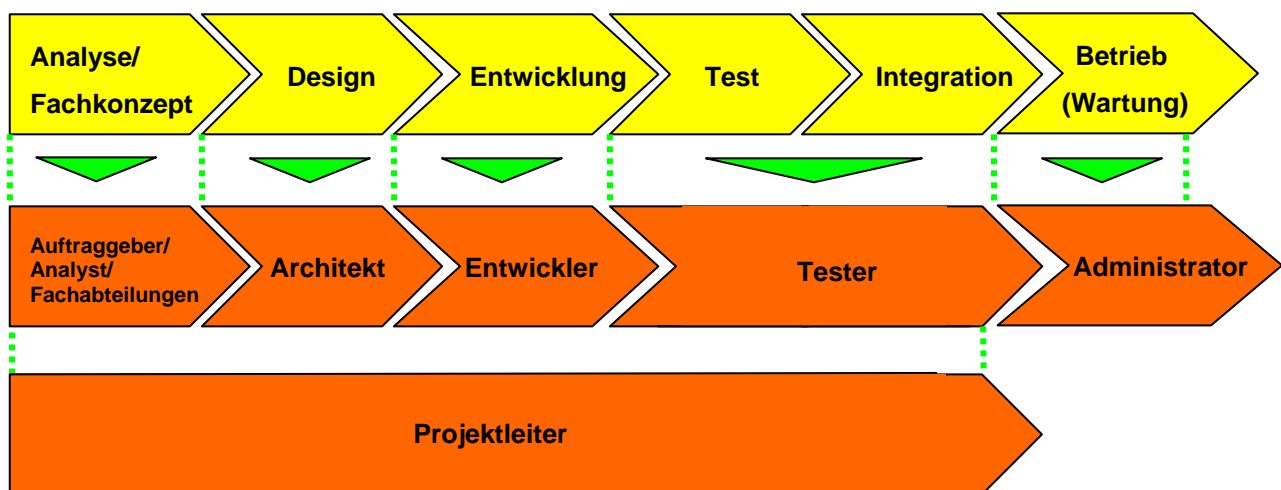


Abb.1: Exemplarische Darstellung der Phasen im Softwarelebenszyklus, sowie die beteiligten Rollen

Wenn von *Zehn Goldenen Regeln* (bzw. *Ten Golden Rules*) im Rahmen von Sicherheit in der Entwicklung von Software gesprochen wird, wurden häufig Regeln für den Software-Entwickler beschrieben. Da aber Sicherheit nicht als losgelöster, isolierter Prozess zu betrachten ist, sind die oben zugeordneten Rollen alle in unterschiedlicher Art und Weise mit an dem Erstellen sicherer Software beteiligt. Daher haben wir in diesem Dokument für die verschiedenen Rollen jeweils zielgruppenspezifische *10 Goldene Regeln* mit Erklärungen zur Erstellung sicherer Software zusammengefasst.

## Zielgruppen

Die Zielgruppen für dieses Dokument sind alle Personen – sowohl in mittelständischen Unternehmen als auch in Großunternehmen – die Software erstellen bzw. erstellen lassen wollen und den Aspekt Sicherheit aktiv in ihre Betrachtungen mit einbeziehen wollen.

## Überblick der 10 Goldenen Regeln für die beteiligten, unterschiedlichen Rollen im Softwarelebenszyklus

Nr.	Analyst/Auftraggeber	Planer (Architekt)	Entwickler	Tester	Projektleiter
1	Denken Sie über geschäftliche Konsequenzen von IT-Sicherheitsfragen nach	Nutzen Sie die Erfahrung von anderen: <ul style="list-style-type: none"> <li>• Erfinden Sie Schwachstellen nicht neu</li> <li>• Entwickeln Sie keine eigenen Sicherheitsfunktionen!</li> <li>• Internationale Sicherheitsstandards (technische Standards)</li> </ul>	Misstrauen Sie jeder (Benutzer-) Eingabe (Validate Input)	Testen Sie priorisiert Hauptangriffspunkte	Sie sind verantwortlich für die Umsetzung aller Anforderungen, lesen Sie zunächst also alle anderen "Goldenen Regeln"
2	Berücksichtigen Sie gesetzliche und regulatorische Anforderungen	„Minimale Rechte“-Prinzip, Berechtigungskonzept erstellen	„Minimale Rechte“-Prinzip	Machen Sie Code-Reviews	Fordern Sie die fachlichen Sicherheitsanforderungen ein
3	Definieren Sie fachliche Rollen, Zuständigkeiten und Prozesse	Trust Zones: Welchen eingehenden Daten darf vertraut werden und welchen nicht	Keine unnötige Veröffentlichung von internen Daten! (Sanitize Output)	Machen Sie Regressionstests (einmal behobene Fehler sollen zukünftig nicht mehr auftauchen)	IT-Sicherheit gleich von Beginn an, nicht erst am Ende oder hinterher
4	Berücksichtigen Sie auch Anforderungen an die Nachvollziehbarkeit	Machen Sie es einfach!	Nutzen Sie die Erfahrung von anderen (do's und dont's) <ul style="list-style-type: none"> <li>• Erfinden Sie Schwachstellen nicht neu</li> <li>• Entwickeln Sie keine eigenen Sicherheitsfunktionen!</li> </ul>	Binden Sie externe Sicherheitsexperten mit ein (für Schwachstellentest und/oder -reviews)	In allen Projektphasen gibt es Arbeitspakete für IT-Sicherheit
5	Verfügbarkeit und Business Continuity können spezielle Anforderungen sein	Niemals „Security by Obscurity“	Sichere Einstellungen <ul style="list-style-type: none"> <li>• Sichere Voreinstellungen</li> <li>• Alle Benutzerkennungen, Passwörter und Verschlüsselungsschlüssel änderbar!</li> <li>• Machen Sie es einfach!</li> <li>• Sicherheitsdokumentation</li> </ul>	Nutzen Sie vorhandene Testfälle und Tools	Planen Sie hinreichende Ressourcen für IT-Sicherheit
6	Sicherheitsanforderungen sollten bekannt, vereinbart und fixiert sein	Sichere Voreinstellungen	Niemals „Security by Obscurity“ und niemals „Hintertüren“ <ul style="list-style-type: none"> <li>• „Security by Obscurity“</li> <li>• Keine „Hintertüren“</li> </ul>	Testen Sie funktionale Sicherheitsanforderungen	Vergessen Sie nicht Ihre (Projekt-) Mitarbeiter, planen Sie Sensibilisierung und Qualifizierung
7	Die Erfüllung von Sicherheitsanforderungen sollte abnahmerelevant sein	Machen Sie ein Architekturreview	Gute Fehlerbehandlung (Fail securely)	Testen Sie auf Schwachstellen	Nutzen Sie Standards, sichere Komponenten und existierende Erfahrungen
8	Beachten Sie Restrisiken und treffen Sie ggf. Vorkehrungen	Mehrere Sicherheitshürden (Defense in Depth)	Denken Sie an Nachvollziehbarkeit (Trace, Audit, Logs)	Erstellen Sie einen Testplan	Bewahren Sie Handlungsspielraum für mögliche Veränderungen
9	Vergessen Sie nicht Ihre eigenen Prozesse, organisatorischen Maßnahmen und Leute	Rechtliche Anforderungen beachten <ul style="list-style-type: none"> <li>• Datenschutz ermöglichen</li> <li>• Machen Sie alles nachvollziehbar (sofern sinnvoll)!</li> </ul>	Machen Sie Code-Reviews	Führen Sie Sicherheitstests nicht gleichzeitig mit fachlichen Tests durch	Machen Sie Restrisiken transparent und lassen sich diese abzeichnen
10	Alles kann sich ändern, seien Sie darauf vorbereitet	Rechnen Sie mit Schwachstellen: sehen Sie Patches vor	Rechnen Sie mit Schwachstellen: sehen Sie Patches vor	Sehen Sie für alle ("normalen") Testfälle Negativbeispiele vor	Denken Sie an diejenigen, die später mit dem Produkt umgehen müssen

# 1 Die 10 Goldenen Sicherheits-Regeln für „Auftraggeber“ (Fachabteilungen/Analysten)

Am Anfang des Softwarelebenszyklus wird immer nach dem „Was“ gefragt, d.h.: was soll die zukünftige Software leisten? Die Bezeichnung der ausführenden Rolle ist in den unterschiedlichen Branchen uneinheitlich. In diesem Dokument wird diese Rolle den Bezeichnungen *Auftraggeber* oder *Fachabteilung* oder *Analyst* zugeordnet.

## 1.1 Denken Sie über geschäftliche Konsequenzen von IT-Sicherheitsfragen nach (Think about Business Impact of Potential Security Breaches)

Wesentliche Sicherheitsanforderungen sind z.B. Vertraulichkeit und Integrität der Informationen und Daten, sowie die Anforderung, dass nur berechtigte Personen auf bestimmte Informationen zugreifen dürfen. Verletzungen solcher Anforderungen durch nicht implementierte, eigentlich aber erforderliche, Sicherheitsmaßnahmen können zu erheblichen geschäftlichen Schäden führen. Sicherheit ist aber auch kein Selbstzweck und Sicherheitsmaßnahmen kosten Zeit und Geld, überflüssige oder übertriebene Maßnahmen können also auch unnötigen Aufwand verursachen.

Entscheidender Maßstab sind also die geschäftlichen Anforderungen („Business Requirements“) an die IT-Sicherheit, für die dann entsprechende Sicherheitsmaßnahmen zu implementieren sind. Aus fachlicher Sicht lassen sich diese am besten in der Form der Betrachtung der geschäftlichen Konsequenzen („Business Impact“) ableiten. Geeignete Fragestellungen, um diese Anforderungen zu ermitteln lauten:

- „Was wäre wenn“ bestimmte Sicherheitsziele verletzt sind oder werden?
- „Welcher Schaden könnte eintreten wenn“ Sicherheitsziele verletzt sind oder werden?

Nicht vergessen werden sollten dabei auch Imageschäden, die allerdings schwer zu quantifizieren sind.

Im Mittelpunkt stehen also mögliche fachliche bzw. monetäre Konsequenzen bei Verletzung der Sicherheitsziele, zu betrachten sind dabei u.a.:

- die betroffenen Informationen,
- mögliche Szenarien und Bedrohungen,
- eventuelle Geschädigte sowie
- „Nutznießer“ bzw. „Interessenten“.

Dabei muss man aber realistisch bleiben, ein Anspruch auf 100%ige Sicherheit ist kostenintensiv und in der Praxis ohnehin unerfüllbar.

Beispiele für solche Fragestellungen wären:

- Was wäre, wenn der Konkurrenz die Einkaufskonditionen bekannt werden ?
- Ab welcher Zeit kann der Ausfall des System unternehmenskritisch werden (Regel 1.5) ?

## 1.2 Berücksichtigen Sie gesetzliche und regulatorische Anforderungen (Check Legal and Regulatory Requirements)

Unabhängig von den geschäftsgetriebenen Sicherheitsanforderungen (siehe Regel 1.1) kann es darüber hinaus gehende gesetzliche oder regulatorische Sicherheitsanforderungen geben. Beispiele hierfür sind Datenschutzbestimmungen, Aufzeichnungs- und Aufbewahrungspflichten, Vorgaben bei Outsourcing, Einschränkungen beim Einsatz bestimmter Technologien (z.B. Verschlüsselung), spezielle Anforderungen zuständiger Aufsichtsbehörden. Gesetzliche und regulatorische Vorgaben sind in der Regel national unterschiedlich, bei Einsatz im Ausland müssen also auch die jeweiligen landesspezifischen Anforderungen betrachtet werden.

Bei Unklarheiten hinsichtlich gesetzlicher Anforderungen sollte unbedingt juristische Expertise hinzugezogen werden, da gravierende Verletzungen gesetzlicher Vorgaben weit reichende Folgen haben können.

## 1.3 Definieren Sie fachliche Rollen, Zuständigkeiten und Prozesse (Define business roles, responsibilities and processes)

Letztlich sind die zu erstellenden Programme und die damit zu verarbeitenden Informationen dazu da, Geschäftsprozesse zu unterstützen. Dabei müssen sie mit anderen Programmen sowie fachlichen Nutzern interagieren können. Eine wichtige Grundlage für einen Großteil der Sicherheitsziele besteht aus der Frage, welche Informationen welchen Personenkreisen mit welchen Rechten zur Verfügung stehen dürfen.

Bei hohen Sicherheitsanforderungen gilt hier oft das „Need to Know“-Prinzip. Dieser Grundsatz besagt, dass nur diejenigen Personen auf nur die für sie wichtigen Informationen mit den genau dafür nötigen Rechten zugreifen dürfen, die sie für ihre tatsächliche Arbeit benötigen. Bei hohen Sicherheitsanforderungen muss dies in der Regel technisch unterstützt sein, d.h. dass durch entsprechende Sicherheitsmaßnahmen ausgeschlossen werden kann, dass Unberechtigte auf nicht für sie bestimmte Informationen zugreifen können.

Fachlich gesehen sind je nach Aufgabenstellung und Einsatzzweck die unterschiedlichsten Rollen denkbar, die mit unterschiedlichen Rechten (z.B. keine, lesend, schreibend, usw.) auf die zugehörigen Informationen zugreifen dürfen.

Beispiele für u.U. zu trennende fachliche Rollen: Auftragserfassung, Auftragsgenehmigung, Auftragsfreigabe, Inkasso, unabhängige Kontrolle, usw.. Der IT dienen diese Informationen dazu, geeignete „Rollen- und Berechtigungskonzepte“ vorzusehen und zu implementieren, und damit entsprechende Vorkehrungen zu treffen, dass tatsächlich nur die jeweiligen Berechtigten auch über die erforderlichen Zugriffsrechte verfügen.

## 1.4 Berücksichtigen Sie auch Anforderungen an die Nachvollziehbarkeit (Consider also “Traceability” Requirements)

Über die “herkömmlichen” Sicherheitsanforderungen hinaus (siehe vorhergehende Regeln) kann es oft spezielle Sicherheitsanforderungen dahingehend geben, in wie weit Ergebnisse und Einzelschritte aus der Vergangenheit nachvollziehbar sein müssen.

Beispiele:

- Typische Fragestellung einer internen Revision, einer externen Aufsichtsbehörde oder aber auch der Geschäftsleitung kann sein: „Wer hatte denn vor 7 Monaten mit welchen Rechten Zugriff auf die Personaldaten und wer hat was genau am Tag XY gemacht“.
- Rechtliche Anforderungen an die Nachvollziehbarkeit bestimmter Sachverhalte und Informationen, z.B. hinsichtlich bilanzrelevanter Daten etc. (vgl. Regel 1.2).
- Nachvollziehbarkeit von Zugriffen auf personenbezogene Daten (lesen/schreiben) aufgrund von Datenschutzbestimmungen.

Derlei Anforderungen an die Nachvollziehbarkeit müssen identifiziert und definiert werden, so dass die IT-Verantwortlichen daraus ein „Logging-“ und „Archivierungskonzept“ ableiten können. Bei bestimmten hochkritischen Log-Dateien kann es darüber hinaus sogar der Fall sein, dass bestimmte Aufzeichnungen und Log-Dateien speziell gegen Veränderungen geschützt werden müssen.

Zu beachten ist, dass es aus Datenschutzgründen heraus auch Löschanforderungen geben kann, die die Aufbewahrung personenbezogener Daten begrenzen (vgl. Regel 1.2).

## 1.5 Verfügbarkeit und Business Continuity können spezielle Anforderungen sein (Availability and Business Continuity can be special Requirements)

Beim Themenkomplex Verfügbarkeit sind zwei wesentliche Arten von Sicherheitsanforderungen zu unterscheiden.

Einerseits kann es um die Frage gehen, wie lange ein Unternehmen temporär ohne bestimmte Anwendungen und Informationen seinen Geschäftsbetrieb aufrecht erhalten kann, bzw. welche Schäden temporäre Nichtverfügbarkeiten nach sich ziehen können. Hier gilt es also z.B. zu überlegen welche „Wiederanlaufzeiten“ (z.B. bei Ausfällen bestimmter Komponenten) minimal vorzusehen sind. Diese Anforderungen erlauben der IT entsprechende Vorkehrungen zu treffen (z.B. USV) oder fließen in Wartungs- und Supportverträge ein. Höhere Anforderungen führen jedoch natürlich auch zu höheren Kosten, es ist daher wichtig, den geeigneten Kompromiss zu finden.

Andererseits existiert auch das potentielle Risiko des dauerhaften Verlustes bestimmter Daten. Der Totalverlust unverzichtbarer Informationen kann im schlimmsten Fall sogar den Fortbestand eines Unternehmens insgesamt gefährden. Darüber hinaus sollte auch ein Katastrophenfall (z.B. komplette Überflutung des Rechenzentrums) ins Kalkül gezogen werden. Je nach Anforderungen sollte also überlegt werden, welche Maßnahmen hier vorzusehen sind (z.B. wöchentliche komplette Datensicherung und Lagerung an einem anderen Ort oder gar Betrieb einer separaten Notfalllokation oder an einem anderen Standort des Unternehmens). In einem Notfallplan sollte darüber hinaus möglichst konkret geregelt werden,

was in einem solchen Katastrophenfall zu tun ist, wer zu informieren ist, wer Entscheidungen trifft usw. Sinnvollerweise werden solche Notfallpläne durch regelmäßige Tests untermauert.

Allgemein sei noch erwähnt, daß in gewissen Branchen (z.B. Finanzdienstleistungssektor) sogar gesetzliche und regulatorische Anforderungen an Notfallmaßnahmen und „Business Continuity“ existieren.

## **1.6 Sicherheitsanforderungen sollten bekannt, vereinbart und fixiert sein (Security Requirements should be known, agreed and signed off)**

Die Umsetzung gewisser Sicherheitsanforderungen wird seitens des Auftraggebers oft stillschweigend vorausgesetzt, umso größer ist hinterher die Überraschung im Sinne „Wie konnte denn das passieren“ bei etwaigen Sicherheitsverletzungen. Seitens des fachlichen Auftraggebers kann jedoch nicht davon ausgegangen werden, dass IT-Mitarbeitern die fachliche Bedeutung bestimmter Daten immer klar ist. Letztlich sind es primär die fachlichen Verantwortlichen, die ihr Geschäft kennen müssen und somit beurteilen können, welche Schäden und Konsequenzen es haben kann, wenn bestimmte Sicherheitsziele verletzt werden. Es kann z.B. nicht stillschweigend vorausgesetzt werden, dass IT-Spezialisten alle Gesetze kennen, oder dass ihnen klar ist, wie sensibel Informationen wie z.B. Einkaufskonditionen sein können.

Wichtige Voraussetzung ist daher, dass die Sicherheitsanforderungen allen relevanten Beteiligten bekannt sind und dass sie – wie andere „normale“ Anforderungen auch – schriftlich festgeschrieben sind. Je nach Konstellation ist das entsprechende Dokument z.B. das „Fachkonzept“, „Pflichten“- , „Lasten“- oder „Anforderungsheft“ oder der abgeschlossene Vertrag. Aufgabe des Auftraggebers ist es dabei die fachlichen Anforderungen an die Sicherheit zu formulieren und zu fixieren. Die konkrete technische Auskleidung der dafür nötigen Sicherheitsmaßnahmen ist Aufgabe der beteiligten IT-Verantwortlichen.

Sind die Sicherheitsanforderungen des fachlichen Auftraggebers fixiert, so dass die IT-Verantwortlichen diese berücksichtigen müssen, kann es in der Praxis durchaus dazu kommen, dass weiterer Abstimmungsbedarf besteht, z.B. wenn Preis/Aufwand für bestimmte Sicherheitsmaßnahmen gegen den Nutzen (bzw. damit zu verhindernde Schäden) abgewogen werden müssen. Diese Diskussionen zwischen Auftraggebern und den beteiligten IT-Verantwortlichen können aber sehr förderlich sein, und führen häufig zu Sicherheitsmaßnahmen mit besserem Kosten/Nutzen-Verhältnis. Wichtig dabei ist, dass diese Klärungen zu einem frühen Zeitpunkt erfolgen sollten, z.B. wenn das Fachkonzept oder der Vertrag erarbeitet wird. Nachträglich lassen sich solche Anforderungen nur schwer und mit großem Aufwand berücksichtigen.

## **1.7 Die Erfüllung von Sicherheitsanforderungen sollte abnahmerelevant sein (Fullfilment of Security requirements should be subject of acceptance)**

Für Sicherheitsanforderungen gilt der gleiche Grundsatz wie für „herkömmliche“ fachliche Anforderungen, die Erfüllung der Anforderungen sollte relevant für die Abnahme des Ergebnisses sein. Im Zweifel, bei Uneinigkeit oder bei Nichterfüllung sind es letztlich die vorher (schriftlich) vereinbarten Abnahmekriterien, die darüber entscheiden, ob das Gesamtergebnis abgenommen werden muss oder ob die Abnahme verweigert werden kann und Nachbesserungsbedarf besteht. Dies gilt vor allem bei externen Auftraggebern, im Zweifel entscheidet hier nur der schriftliche Vertrag.

Für einen IT-sicherheitstechnisch weniger versierten Auftraggeber kann es allerdings schwierig sein, den realen Erfüllungsgrad von Sicherheitsanforderungen tatsächlich ermitteln zu können. Für gewisse Themen (z.B. Rollenkonzept) ist eine Verifikation eher möglich, für andere Themen (z.B. Inputvalidierung) ist der Auftraggeber eventuell technisch überfordert. Dennoch ist es auch hierfür sinnvoll, entsprechende Kriterien zu formulieren, und sei es um späteren Nachbesserungsbedarf untermauern zu können. Bzgl. der Sicherheit von Anwendungssoftware kann man z.B. auf die öffentlich verfügbaren „Top Ten“ von OWASP zurückgreifen und vereinbaren, dass ein Auftauchen einer dort aufgeführten „Vulnerabilities“ einen Mangel darstellt.

Bei hochkritischen Anwendungen kann es außerdem sinnvoll sein, eine externe (d.h. von der beauftragten IT unabhängige) Prüfung zur Abnahme des Ergebnisses zu beauftragen. Soweit sinnvoll und erforderlich, sollte dies (außer in begründeten Ausnahmefällen) aber möglichst im Einklang mit den Auftragnehmern geschehen. D.h. wenn dies schon vorher klar ist, sollte es auch schon vorher angekündigt sein, dass eine solche Abnahme geplant ist. Wird die Notwendigkeit einer unabhängigen Prüfung erst später klar, dann sollte dies möglichst unter Mitwirkung und Beteiligung der Auftragnehmer geschehen.

Gründe für eine solchermaßen abgestimmte Vorgehensweise gibt es mehrere, z.B.:

- Ein hoffentlich vorhandenes Vertrauensverhältnis mit dem Ersteller sollte nicht unnötig gefährdet werden, üblicherweise handelt es sich ja um eine längere Beziehung.
- Gefundene Schwachstellen sollen letztlich behoben werden, typischerweise durch denjenigen, der die Software am besten kennt, also den Ersteller.
- In Kenntnis bereits geplanter unabhängiger Überprüfungen bemüht sich der Auftragnehmer u.U. von vorneherein mehr um das Thema Sicherheit.

## 1.8 Beachten Sie Restrisiken und treffen Sie ggf. Vorkehrungen (Be aware of remaining security risks and be prepared)

100%ige Sicherheit ist auch bei noch so hohem Aufwand nicht darstellbar, und auch Wirtschaftlichkeitsüberlegungen können dazu führen, dass gewisse, zwar technisch mögliche, aber zu aufwendige Sicherheitsmassnahmen nicht umgesetzt werden. Wichtig ist dann aber, sich über die verbleibenden Restrisiken im Klaren zu sein.

Bei kritischen Anwendungen sollte am besten bereits vorab mit den IT-Verantwortlichen vereinbart und fixiert sein, dass vor Abnahme (d.h. vor Projektende oder Auslieferung) eine Aufstellung der nicht oder nur eingeschränkt durch IT-technische Sicherheitsmaßnahmen abdeckbaren Restrisiken erfolgen soll.

Der Auftraggeber sollte bei relevanten verbleibenden Restrisiken überlegen, welche Maßnahmen bei Eintritt der zugrunde liegenden Szenarien getroffen werden können oder welche flankierenden nicht-technischen Maßnahmen Abhilfe schaffen könnten.

(Beispiele: Zusätzliche organisatorische Maßnahmen wie bestimmte Arbeitsanweisungen oder zusätzliche Archivierung von papierhaften Belegen)

## 1.9 Vergessen Sie nicht Ihre eigenen Prozesse, organisatorischen Maßnahmen und Leute (Don't forget your own Processes, Organizational Measures and People)

Nicht alle Sicherheitsmaßnahmen sind Sache der IT-Verantwortlichen. So mag die IT zwar ein den fachlichen Anforderungen entsprechendes, technisch ausgeklügeltes Berechtigungskonzept entwickelt haben, wenn aber später (nachdem das Projekt längst zu Ende ist und das Produkt sich im laufenden Betrieb befindet) User-IDs und Passwörter ohne Überprüfung der tatsächlichen Rolle und Berechtigung vergeben werden, war der ganze technische Aufwand möglicherweise völlig umsonst. Ähnliches gilt, wenn nicht gewährleistet ist, dass z.B. ausgeschiedene Nutzer gelöscht werden. Daher sind die eigenen Prozesse schon in der Planung zu berücksichtigen sowie entsprechende organisatorische Maßnahmen (z.B. Antragsverfahren und entsprechende Formulare) zu definieren. Hierbei sollte spezifiziert werden wer auf welcher Basis und mit welchen Genehmigungsschritten welche Zugriffsrechte haben darf. Dies muss möglicherweise mit Sensibilisierungs-Maßnahmen kombiniert werden, damit die definierten Prozesse auch wirklich eingehalten werden.

Darüber hinaus dürfen nicht diejenigen vergessen werden, die später mit den Daten, Programmen und mit den damit verbundenen Sicherheitsmaßnahmen umgehen müssen. Oft sind auch hier flankierende Sensibilisierungs- oder gar Schulungsmaßnahmen vorzusehen.

## 1.10 Alles kann sich ändern, seien Sie darauf vorbereitet (Everything can be subject to change, be prepared)

Unter dieser Regel lassen sich verschiedene mögliche Veränderungen (während des Projekts, vor allem aber auch nach dem Projektablauf) zusammenfassen:

Treffen Sie Regelungen für erst später offensichtlich werdende Mängel:

Auch wenn es am Ende eines Projektes so aussieht, als ob alle Sicherheitsanforderungen umgesetzt sind, so ist es in der Praxis häufig der Fall, dass sich erst später im laufenden Betrieb herausstellt, dass gewisse Sicherheitsanforderungen nur teilweise umgesetzt sind oder dass gewisse Mängel erst nachträglich festgestellt werden. Darüber hinaus muss auch die heutzutage leider schon routinemäßig vorzusehenden Einspielung von Sicherheits-Updates für Standard-Software wie z.B. Betriebssysteme etc. berücksichtigt werden. Hier werden regelmäßig neue Sicherheitslücken entdeckt, die seitens des Herstellers dann durch „Security Patches“ behoben werden.

Es ist daher wichtig, bereits vorab Maßnahmen zur Behebung möglicher Mängel vorzusehen. Sowohl bei intern, insbesondere aber bei extern entwickelten, gekauften oder gar extern betriebenen Anwendungen sollten Nachbesserungspflichten auch für Sicherheitsanforderungen in Wartungs- und Supportvereinbarungen festgehalten sein (Stichwort „Service Level Agreement“).

Halten Sie sich auf dem Laufenden:

Neue fachliche Gegebenheiten können Veränderungen an Sicherheitsanforderungen (schon während der Projektlaufzeit, aber auch viel später im laufenden Betrieb) nach sich ziehen. Zu nennen sind z.B. neue gesetzliche Vorgaben, Neubewertungen der fachlichen Sicherheitsanforderungen (z.B. nach „nine-eleven“) usw. Man kann jedoch nicht davon ausgehen, dass die IT solche fachlich begründeten Neuanforderungen von selbst wahrnimmt oder gar aktiv und eigenständig berücksichtigt und umsetzt. Im Zweifel muss die Umsetzung solcher geänderter Sicherheitsanforderungen kommuniziert oder gar separat beauftragt werden.

## 2 Die 10 Goldenen Sicherheits-Regeln für „Planer“ (Architekten)

In der Phase Design des Softwarelebenszyklus wird immer nach dem WIE gefragt, d.h. wie soll die zukünftige Software die Anforderungen aus der Analyse erfüllen. Die Bezeichnung der ausführenden Rolle ist in den unterschiedlichen Branchen uneinheitlich. In diesem Dokument wird diese Rolle den Bezeichnungen *Architekt* oder *Planer* zugeordnet.

### 2.1 Nutzen Sie die Erfahrung von anderen

Andere Entwicklungsprojekte haben schon eine ganze Reihe Erfahrungen gemacht. Viele dieser Erfahrungen sind käuflich erhältlich oder sogar frei verfügbar. Ggf. liegen entsprechende Erfahrungen sogar im eigenen Unternehmen vor. Benutzen Sie diese Erfahrungen und machen Sie nicht die gleichen Fehler, die andere schon gemacht haben.

#### 2.1.1 Erfinden Sie Schwachstellen nicht neu

Gute Startpunkte für Tipps zum sicheren Programmieren:

- Microsoft Security Developer Center, SAP Developer Network (dort den NetWeaver Developer Guide)
- Das BSI, dort z.B. das Grundschutzhandbuch mit dem Abschnitt zu sicherem Programmieren (ist in Vorbereitung)
- Open Web Application Security Project (dort insbesondere den OWASP Guide)
- Andere "Secologic" Dokumente

#### 2.1.2 Entwickeln Sie keine eigenen Sicherheitsfunktionen!

Sofern Sie keine "Sicherheitsanwendung" bauen, werden Ihre Sicherheitsfunktionen sowieso kein Alleinstellungsmerkmal ihrer Anwendung sein. Verlassen Sie sich darum auf die Arbeitsergebnisse von Sicherheitsspezialisten für das jeweilige Themengebiet. Sowohl im kommerziellen als auch im freien Umfeld gibt es ausreichend getestete Funktionen/Bibliotheken.

Typische Sicherheitsfunktionen in diesem Sinne sind Funktionen z.B. für:

- Verschlüsselung
- Anmeldung
- Berechtigungsverwaltung
- Digitale Signatur

#### 2.1.3 Internationale Sicherheitsstandards (technische Standards)

Setzen Sie bei der Verwendung der jeweiligen Technologie existierende Sicherheits-Standards und Empfehlungen ein.

Beispiele hierfür wären:

- Bei der Verwendung von Web Services: Einsatz von Standards wie u.a. WS-Security und SAML.
- Bei der Verwendung einer sicheren Ende zu Ende Kommunikation: Nutzung einer SSL Verschlüsselung.

### 2.2 „Minimale Rechte“-Prinzip, Berechtigungskonzept erstellen

Geben Sie Ihren Kunden die Möglichkeit, Benutzern stets nur die Berechtigungen zu geben, die sie zur Erfüllung ihrer Aufgabe wirklich brauchen. Zu viele Rechte sind ein Sicherheitsrisiko.

Denken Sie auch an die Aufgaben für Installation/Konfiguration und Administration. Gibt es unterschiedliche Administrationsaufgaben?

Das minimale Rechteprinzip gilt auch für:

- Technische Benutzer
- Betriebssystem- /Datenbankbenutzer (eine Anwendung sollte z.B. nicht mit Administrator-/"Root"/"System"-Rechten laufen)
- Zugriffsberechtigungen im Dateisystem, das heißt zum Beispiel:
  - Konfigurationsdateien sollen nicht von jedem änderbar sein.
  - Dateien mit vertraulichem Inhalt dürfen nur von berechtigten Benutzern zugreifbar sein.

Vorsicht auch, wenn auf Daten oder Funktionen über mehrere Wege zugegriffen werden kann. Dann sollten trotzdem stets die gleichen Zugriffsregeln gelten. Das gilt übrigens auch, wenn man Daten an einen anderen Ort kopiert (z.B. für Test- oder Auswertungszwecke).

### 2.3 Trust Zones: Welchen eingehenden Daten darf vertraut werden und welchen nicht

Jede Funktion ist für die Überprüfung sämtlicher Eingaben selbstverantwortlich. Zur Vereinfachung sollte man Funktionsgruppen definieren, die ein gemeinsames Sicherheitsniveau versprechen. Innerhalb einer solchen Gruppe können sich alle anderen Funktionen auf das Sicherheitsniveau verlassen.

Dafür müssen eingehende Daten an den Außen-Schnittstellen einer solchen Gruppe (Trust Zone) besonders gründlich geprüft werden.

### 2.4 Machen Sie es einfach!

Nur wenn ihre Sicherheitseinstellungen und -funktionen einfach zu bedienen sind wird der Kunde diese auch nutzen. Machen Sie es ihm darum einfach, z.B. durch:

- Warnungen bei Einstellungen, die zu Sicherheitslücken führen
- Assistenten zum einfachen Einstellen
- Lassen Sie unsichere Konfigurationen gar nicht zu, wenn es nicht notwendig ist

### 2.5 Niemals „Security by Obscurity“

Vertrauen Sie niemals darauf, dass Ihre Software nicht angreifbar ist, nur weil der Quellcode bzw. die Sicherheitsmechanismen unbekannt sind.

Einerseits wird irgendwann jemand sowieso dahinter kommen. Wenn man dann keine gute Lösung hat, kann es sehr unangenehm werden. Gehen Sie immer davon aus, dass die Software auch dann noch gut sein muss, wenn der Quellcode bekannt würde.

Andererseits sind „obskure“ Lösungen häufig auch intern nicht dokumentiert und es gibt langfristig Schwierigkeiten mit der Wartung und Wiederverwendung.

Typisches Beispiel sind „selbst erfundene“ Verschlüsselungsverfahren.

### 2.6 Sichere Voreinstellungen

Nicht jeder Ihrer Kunden wird alle Ihre Auslieferungsdokumentation lesen. Darum wird er evtl. nicht erfahren, dass es sicherere Einstellungen gibt. Sorgen Sie dafür, dass die Anwendung nach der Installation bereits in einem sicheren Zustand ist. Alternativ kann für den Testbetrieb ein unsicherer

Zustand akzeptiert werden, wenn der Kunde eine einfache Möglichkeit hat, vor dem Produktivstart die kritischen Einstellungen einfach in einen sicheren Zustand bringen.

Zu einem sicheren Zustand gehört auch, dass es keine unnötigen offenen Ports bzw. Kommunikationsdienste gibt und dass unnötige Funktionen deaktiviert sind (->Angriffsfläche reduzieren). Der Benutzer sollte aufgefordert werden, vorgelegte Passwörter zu ändern.

## 2.7 Machen Sie ein Architekturreview

Planen Sie als Entwicklungsschritt Architektur-Reviews durch einen Kollegen fest ein. Diese Architektur-Reviews haben 2 wesentliche Vorteile:

- Es gibt eine 2. Meinung zur vorgeschlagenen Lösung. Ohne Vorprägung durch sehr intensive Vorüberlegungen kann ein "unabhängiger" Kollege Sicherheitsschwachstellen häufig einfacher finden.
- Die Qualität der Architektur verbessert sich nach und nach, weil jeder Architekt weiß, dass:
  - Seine Spezifikation die Reviews durch andere "überleben" muss und
  - jeder aus den Fehlern von Kollegen lernt und darum nach und nach eine nachvollziehbarere, sichere Spezifikation schreibt.

Lassen Sie im Zweifelsfall oder bei besonders kritischen Anwendungen auch externe Spezialisten einen Review machen.

## 2.8 Mehrere Sicherheitshürden (Defense in Depth)

Machen Sie es dem Angreifer schwer, der Aufwand für einen Angriff muss unattraktiv hoch sein. Dies erreichen Sie indem sich Ihre Anwendung nie nur auf einen einzigen Sicherheitsmechanismus verlässt. Es ist immer besser, mehrere Hürden aufzubauen. Dies Vorgehen erhöht den Schutz sowohl vor unerwarteten (neuen) Angriffsmustern als auch die Sicherheit bei späteren Erweiterungen der Software.

Und nicht vergessen: eine Hürde heißt „Nachvollziehbarkeit“, damit der Angreifer wenigstens nachträglich überführt werden kann.

## 2.9 Rechtliche Anforderungen beachten

Die Software soll den Anwender bei seiner Aufgabenerfüllung unterstützen. Diese Aufgaben unterliegen im geschäftlichen Bereich diversen rechtlichen Auflagen. Dazu gehören u.a. Vorgaben zu Nachvollziehbarkeit, Aufbewahrungsfristen, Risikomanagement und Datenschutz. Fragen Sie Ihren Kunden konkret nach diesen Anforderungen und planen diese mit ein.

### 2.9.1 Datenschutz ermöglichen

Die meisten Staaten haben Datenschutzgesetze, die dort tätige Unternehmen einhalten müssen. Beschreiben Sie, wie Ihre Kunden die Anwendungen datenschutzgerecht einsetzen können. Damit erweisen Sie Ihren Kunden einen großen Dienst.

Nutzen Sie das Thema Datenschutz, um sich von anderen zu unterscheiden!

### 2.9.2 Machen Sie alles nachvollziehbar (sofern sinnvoll)!

Sowohl für die interne Revision als auch für Steuer- und Wirtschaftsprüfer ist eine ausreichende Nachvollziehbarkeit wichtig. Protokollieren Sie alles, was in der Anwendung an sicherheitskritischen Ereignissen auftritt. Protokollieren Sie z.B. alle Änderungen in der Benutzer- und Berechtigungsverwaltung, sowie wesentliche Ereignisse wie z.B. erfolgreiche und fehlgeschlagene Anmeldeversuche.

Denken Sie auch an die Archivierung dieser Daten. Protokolle, die schon nach kurzer Zeit überschrieben werden, können danach nicht mehr nachvollzogen werden.

Denken Sie aber daran, dass in solchen Protokollen vertrauliche Daten stehen können und sehen Sie Zugriffsberechtigungen vor.

## 2.10 Rechnen Sie mit Schwachstellen: sehen Sie Patches vor

Es können auch nach der Auslieferung der Software neue Bedrohungsszenarien auftreten. Daher sollte man schon bei der Architekturplanung die Möglichkeit des Einspielens von Patches einplanen. Ein modularer Aufbau der Software unterstützt dieses Vorgehen. Es sollte für den Kunden ein nutzerfreundliches Einspielen eines Patches ermöglicht werden. Zudem sollte der Kunde feststellen können, ob er den notwendigen Patch schon eingespielt hat.

## 3 Die 10 Goldenen Sicherheits-Regeln für „Developer“ (Entwickler)

In der Phase der Entwicklung des Softwarelebenszyklus wird die Spezifikation der Architektur in Code umgesetzt. Die Bezeichnung der ausführenden Rolle ist in den unterschiedlichen Branchen einheitlich. Sie ist nur abhängig von dem Gebrauch der Sprache deutsch oder englisch (Developer oder Entwickler).

### 3.1 Misstrauen Sie jeder (Benutzer-) Eingabe (Validate Input)

Jede Eingabe kann unerwartete Inhalte haben. Eingaben können nicht nur von der Benutzeroberfläche kommen – sondern auch aus Dateien und Serviceaufrufen. Ein blindes Vertrauen ist heute die häufigste Sicherheitsschwachstelle in Applikationen.

Ablauf für die Prüfung:

1. Alles in eine kanonische (einheitliche) Form bringen
2. Bekannte Angriffe filtern (Blacklist) (wichtiger Schritt, aber nicht ausreichend!)
3. Nur erwartete Eingaben annehmen (Whitelist)

Einige Grundregeln:

- Filtern Sie unerwünschte Steuerzeichen aus (auch TAB, ;, Hochkomma, LF, NUL, usw).
- Achten Sie wirklich auf jeden extern veränderbaren Wert (auch HTTP- Header, URLs, Cookies, ...)
- Beachten sie unbedingt, ob die Länge der Eingaben mit dem übereinstimmt, was sie erwarten (insbesondere bei Buffer Overflow- problematischen Programmiersprachen wie C/C++)
- Auf alle veränderbaren Eingaben achten (im Webumfeld z.B. auch URLs, Cookies, Hidden Fields, HTTP Header, ...)
- Vertrauen Sie niemals Prüfungen nur im Browser. Browserseitige Prüfungen können einfach manipuliert oder deaktiviert werden. Prüfen sie immer (auch) auf dem Server
- Vorsicht vor Directory Traversal (unerwartete Pfadangaben)

Diese Grundregeln schützen Sie weitgehend vor den typischen Schwachstellen von Internetanwendungen wie:

- Cross Site Scripting
- SQL-Injection
- Buffer Overflows
- Code Injection (z.B. für Betriebssystembefehle)

### 3.2 „Minimale Rechte“-Prinzip

Geben Sie Ihren Kunden die Möglichkeit, Benutzern stets nur die Berechtigungen zu geben, die sie zur Erfüllung ihrer Aufgabe wirklich brauchen. Zu viele Rechte sind ein Sicherheitsrisiko. Sie sollten die Applikation so implementieren, dass jegliche Berechtigungen nicht hart codiert sondern durch den Administrator konfigurierbar sind.

Denken Sie auch an die Aufgaben für Installation/Konfiguration und Administration. Gibt es unterschiedliche Administrationsaufgaben?

Das minimale Rechteprinzip gilt auch für:

- Technische Benutzer
- Betriebssystem- /Datenbankbenutzer (eine Anwendung soll z.B. nicht mit Administrator- /"Root"/"System"-Rechten laufen)
- Zugriffsberechtigungen im Dateisystem, das heißt zum Beispiel:
  - Konfigurationsdateien sollen nicht von jedem änderbar sein.
  - Dateien mit vertraulichem Inhalt dürfen nur von berechtigten Benutzern zugreifbar sein.

Vorsicht auch, wenn auf Daten oder Funktionen über mehrere Wege zugegriffen werden kann. Dann sollten trotzdem stets die gleichen Zugriffsregeln gelten. Das gilt übrigens auch, wenn man Daten an einen anderen Ort kopiert (z.B. für Test- oder Auswertungszwecke).

### 3.3 Keine unnötige Veröffentlichung von internen Daten! (Sanitize Output)

Achten Sie darauf, wer welche Daten zu sehen bekommt.

Während innerhalb der eigentlichen Anwendungsfunktionen häufig gut darauf geachtet wird, werden andere Stellen häufig nicht beachtet. Beispiele für solche Stellen sind:

- Fehlermeldungen mit zu vielen Systeminternas (die nutzen dem Anwender wenig, einem Hacker aber viel!)
- Cookies und URLs
- standardmäßig aktivierte Debug-Funktionen
- Systeminformationen, Patchstand
- Logdateien

### 3.4 Nutzen Sie die Erfahrung von anderen (do's und dont's)

Andere Entwicklungsprojekte haben schon eine ganze Reihe Erfahrungen gemacht. Viele dieser Erfahrungen sind schon von Teamkollegen oder andern Mitarbeitern gemacht worden. Weitere sind käuflich erhältlich oder sogar frei verfügbar. Benutzen Sie diese Erfahrungen und machen Sie nicht die gleichen Fehler, die andere schon gemacht haben.

#### 3.4.1 Erfinden Sie Schwachstellen nicht neu

Gute Startpunkte für Tipps zum sicheren Programmieren:

- Microsoft Security Developer Center, SAP Developer Network (dort den NetWeaver Developer Guide)
- Das BSI, dort z.B. das Grundschutzhandbuch mit dem Abschnitt zu sicherem Programmieren (ist in Vorbereitung)
- Open Web Application Security Project (dort insbesondere den OWASP Guide)
- Andere "Secologic" Dokumente

#### 3.4.2 Entwickeln Sie keine eigenen Sicherheitsfunktionen!

Sofern Sie keine "Sicherheitsanwendung" bauen, werden Ihre Sicherheitsfunktionen sowieso kein Alleinstellungsmerkmal ihrer Anwendung sein. Verlassen Sie sich darum auf die Arbeitsergebnisse von Sicherheitsspezialisten für das jeweilige Themengebiet. Sowohl im kommerziellen als auch im freien Umfeld gibt es ausreichend getestete Funktionen/Bibliotheken.

Typische Sicherheitsfunktionen in diesem Sinne sind Funktionen z.B. für:

- Verschlüsselung
- Anmeldung
- Berechtigungsverwaltung
- Digitale Signatur

## 3.5 Sichere Einstellungen

### 3.5.1 Sichere Voreinstellungen

Nicht jeder Ihrer Kunden wird alle Ihre Auslieferungsdokumentation lesen. Darum wird er evtl. nicht erfahren, dass es sicherere Einstellungen gibt. Sorgen Sie dafür, dass die Anwendung nach der Installation bereits in einem sicheren Zustand ist.

Alternativ kann für den Testbetrieb ein unsicherer Zustand akzeptiert werden, wenn der Kunde eine einfache Möglichkeit hat, vor dem Produktivstart die kritischen Einstellungen einfach in einen sicheren Zustand bringen (Assistent implementieren).

Zu einem sicheren Zustand gehört auch, dass es keine unnötigen offenen Ports bzw. Kommunikationsdienste vorhanden sind und dass unnötige Funktionen deaktiviert sind (→Angriffsfläche reduzieren). Der Benutzer sollte aufgefordert werden, vorgelegte Passwörter zu ändern.

### 3.5.2 Alle Benutzerkennungen, Passwörter und Verschlüsselungsschlüssel änderbar!

Benutzerkennungen, Passwörter und Verschlüsselungsschlüssel schützen den Zugang zu der Anwendung und den vertraulichen Daten. Daher sollten diese bei jeder Anwendungsinstallation geändert werden.

Implementieren Sie niemals festkodierte Passwörter. Diese sind eine Einladung für Missbrauch. Alle Passwörter müssen änderbar sein.

Auch voreingestellte Benutzerkennungen sollten änderbar sein. Grund ist, dass Hacker häufig einen bekannten Benutzernamen als Ansatzpunkt nutzen.

### 3.5.3 Machen Sie es einfach!

Nur wenn ihre Sicherheitseinstellungen und -funktionen einfach zu bedienen sind wird der Kunde diese auch nutzen. Machen Sie es ihm darum einfach, z.B. durch:

- Warnungen bei Einstellungen, die zu Sicherheitslücken führen
- Wizzards(Assistenten) zum einfachen Einstellen
- Lassen Sie unsichere Konfigurationen gar nicht zu, wenn es nicht notwendig ist

### 3.5.4 Sicherheitsdokumentation

Geben Sie dem Anwender die Chance, die Anwendung sicher zu betreiben. Dokumentieren Sie, was dafür notwendig ist. Dokumentieren sie auch vermeintliche Selbstverständlichkeiten (weil es im Bereich Sicherheit keine solchen gibt!).

## 3.6 Niemals „Security by Obscurity“ und niemals „Hintertüren“

### 3.6.1 „Security by Obscurity“

Vertrauen Sie niemals darauf, dass Ihre Software nicht angreifbar ist, nur weil der Quellcode bzw. die Sicherheitsmechanismen unbekannt sind.

Einerseits wird irgendwann jemand sowieso dahinter kommen. Wenn man dann keine gute Lösung hat, kann es sehr unangenehm werden. Gehen Sie immer davon aus, dass die Software auch dann noch gut sein muss, wenn der Quellcode bekannt würde.

Andererseits sind „obskure“ Lösungen häufig auch intern nicht dokumentiert und es gibt langfristig Schwierigkeiten mit der Wartung und Wiederverwendung.

Typisches Beispiel sind selbstgeschriebene Verschlüsselungsverfahren.

### 3.6.2 Keine „Hintertüren“

Jegliche Art von Hintertüren ist unbedingt zu vermeiden. Solche Hintertüren sind z. Bsp. versteckte Funktionalitäten. Diese kommen bei der Softwareentwicklung immer wieder vor und drücken sich in verschiedenen Formen aus.

Beispiele sind:

- versteckte, nicht dokumentierte Funktionalitäten in der Benutzeroberfläche
- versteckte oder undokumentierte Parameter
- undokumentierte Aufrufe über Benutzereingaben

Solche Hintertüren stellen Sollbruchstellen dar, die irgendwann entdeckt werden, nach z. Bsp. einer Veröffentlichung im Internet werden dann gleichzeitig alle Kundensysteme unsicher. Da diese Hintertüren oft weder bekannt noch dokumentiert sind, werden sie bei Tests oder Reviews häufig nicht berücksichtigt.

Ursprünglich unbedenkliche, undokumentierte Hintertüren können in einem anderen Kontext einen gravierenden Sicherheitsmangel auslösen. Dies wäre z. Bsp. bei einer Weiterentwicklung und Wiederverwendung von Funktionen mit Hintertüren der Fall.

## 3.7 Gute Fehlerbehandlung (Fail securely)

Es lassen sich nicht alle möglichen Situationen einer komplexen Anwendungslandschaft vorhersehen oder testen. Darum wird es im Betrieb Fehlersituationen geben. Stellen sie sich auf diese Situationen ein. Stellen Sie vor allem sicher, dass die Anwendung im Fehlerfall in einen sicheren Zustand übergeht (oder ihn behält) und nicht "unsicher" wird.

## 3.8 Denken Sie an Nachvollziehbarkeit (Trace, Audit, Logs)

Protokollieren Sie alles, was in der Anwendung an sicherheitskritischen Ereignissen auftritt. Das sind zum Beispiel:

- Serverstarts und -stopps
- wesentliche Konfigurationsänderungen
- unberechtigte Zugriffsversuche
- Zugriff auf besonders sensible Daten
- alle Änderungen in der Benutzer- und Berechtigungsverwaltung.

Denken Sie auch an die Archivierung dieser Daten. Protokolle, die schon nach kurzer Zeit überschrieben werden, können danach nicht mehr nachvollzogen werden.

Denken Sie daran, dass in Protokollen vertrauliche Daten stehen können und implementieren Sie Zugriffsberechtigungen.

## 3.9 Machen Sie Code-Reviews

Planen Sie als Entwicklungsschritt Code-Reviews durch einen Kollegen fest ein. Diese Code-Reviews haben 2 wesentliche Vorteile:

- Es gibt eine 2. Meinung zur vorgeschlagenen Lösung. Ohne Vorprägung durch sehr intensive Vorüberlegungen kann ein "unabhängiger" Kollege Sicherheitsschwachstellen häufig einfacher finden.
- Die Qualität des Codes verbessert sich nach und nach, weil jeder Entwickler weiß, dass:
  - sein Code die Reviews durch andere "überleben" muss und

- jeder aus den Fehlern von Kollegen lernt und darum nach und nach nachvollziehbareren Code schreibt.

### 3.10 Rechnen Sie mit Schwachstellen: sehen Sie Patches vor

Es können auch nach der Auslieferung der Software neue Bedrohungsszenarien auftreten.

Es sollte für den Kunden ein nutzerfreundliches Einspielen eines Patches ermöglicht werden. Zudem sollte der Kunde feststellen können, ob er den notwendigen Patch schon eingespielt hat.

## 4 Die 10 Goldenen Sicherheits-Regeln für „Tester“

In der Phase des Tests im Softwarelebenszyklus wird die fertige und auch noch in der Implementierung begriffene Software getestet. D.h. der Tester wird nicht erst im Anschluss der Implementierung, sondern auch schon parallel zur Entwicklung tätig.

### 4.1 Testen Sie priorisiert Hauptangriffspunkte

Falls nicht genug Zeit für ein kompletten Test der Anwendung bleibt, konzentrieren Sie sich auf die Hauptangriffspunkte (Priorisierung mit folgender Reihenfolge):

1. Ein- und Ausgabe-Schnittstellen zum Benutzer (z.B. die typischen Angriffspunkte im Webbereich, siehe OWASP Top Ten)
2. Berechtigungsprüfung innerhalb der Anwendung
3. Schnittstellen zu anderen Anwendungen
4. Schnittstellen zum Betriebssystem
5. Schnittstellen zum Dateisystem

### 4.2 Machen Sie Code-Reviews

Code-Reviews sollten durch einen unabhängigen Entwickler/Tester/Experten durchgeführt werden (siehe auch <http://de.wikipedia.org/wiki/Codereview>). Code-Reviews haben 2 wesentliche Vorteile:

- Es gibt eine 2. Meinung zur vorgeschlagenen Lösung. Ohne Vorprägung durch sehr intensive Vorüberlegungen kann ein "unabhängiger" Kollege Sicherheitschwachstellen häufig einfacher finden.
- Die Qualität des Codes verbessert sich nach und nach, weil jeder Entwickler weiß, dass:
  - sein Code die Reviews durch andere "überleben" muss und
  - jeder aus den Fehlern von Kollegen lernt und darum nach und nach nachvollziehbareren Code schreibt.

### 4.3 Machen Sie Regressionstests

Haben Sie bereits beim Testen Schwachstellen gefunden, sollten Sie entsprechende (wenn möglich automatisierte) Tests entwerfen, die für das wiederholte Testen der korrigierten Anwendung eingesetzt werden können. Diese Tests sollten bei allen folgenden Programmversionen möglichst systematisch durchgeführt werden, damit einmal behobene Fehler zukünftig nicht mehr auftauchen.

### 4.4 Binden Sie externe Sicherheitsexperten mit ein (für Schwachstellentest und/oder –reviews)

Für besonders kritische Anwendungen sollten Sie die Unterstützung von externen Sicherheitsexperten in Anspruch nehmen. Diese Spezialisten können durch ihr Wissen über aktuelle Angriffsmuster und typische Fehler und Schwachstellen die Sicherheitstests erheblich anreichern. Neben den oben erwähnten Code Reviews können Sie im Bereich White Box und Black Box Tests unterstützend wirken.

### 4.5 Nutzen Sie vorhandene Testfälle und Tools

Es gibt schon eine Reihe von veröffentlichten Testfällen. Nutzen Sie diese ergänzend zu Ihren applikationsspezifischen Testfällen.

Prüfen Sie ob entsprechend zu Ihrer Entwicklungsumgebung Testtools existieren, die auch Security Testfälle anbieten.

Siehe z.Bsp.:

- <http://hackers.org/xss.html>

- Testfälle aus dem Secologic Projekt

## 4.6 Testen Sie funktionale Sicherheitsanforderungen

Testen Sie, ob die Sicherheitsfunktionen entsprechend der Anforderungen (siehe ‚10 Goldene Regeln‘ des Auftraggebers und des Projektleiters) umgesetzt wurden. Diese Tests sind mit regulären funktionalen Tests vergleichbar und müssen entsprechend der üblichen Kriterien durchgeführt werden.

## 4.7 Testen Sie auf Schwachstellen

Im Gegensatz zu funktionalen Tests haben Sicherheitstests (Test auf Schwachstellen) eine ganz andere Zielsetzung. Funktionale Tests haben die Aufgabe, die Spezifikationen einer Anwendung zu verifizieren, um sicherzustellen, dass die Anwendung sich genau so verhält, wie es gefordert wurde. Beispiel: Alle registrierten Benutzer sollen sich am System anmelden können.

Im Fall der Sicherheitstests ist die Zielsetzung umgekehrt. Sicherheitstests sollen beweisen, dass keine Funktionen vorhanden sind, die nicht vorhanden sein sollen. Beispiel: Alle nicht registrierten Benutzer dürfen sich auch nicht am System anmelden können. Sicherheitstests versuchen somit zu beweisen, dass keine unsicheren Seiteneffekte existieren, damit die Software auch dann noch richtig funktioniert, wenn sie gezielten Angriffen ausgesetzt ist. Sicherheitstests sind jedoch nicht mit funktionalen Negativ-Tests zu verwechseln, die beispielsweise das Auftreten von Fehlermeldungen überprüfen, weil in diesem Fall ebenfalls getestet wird, ob eine bestimmte Funktionalität (das Auftreten der Fehlermeldung) gegeben ist.

## 4.8 Erstellen Sie einen Testplan

Vernünftig konzipierte Sicherheitstests sind reguläre planbare Tests, die nicht nur planloses „Hacking“ beinhalten. Erstellen Sie einen Testplan mit Testfällen wie auch bei regulären funktionalen Tests. Bei einem solchen Vorgehen analog der Vorgehensweise bei herkömmlichen funktionalen Tests, haben Sie die Möglichkeit systematisch einen großen Teil der Schwachstellen bereits frühzeitig zu identifizieren (und ggf. zu beheben). Beispiel: Analog zu Testfällen, bei denen getestet werden soll, ob sich registrierte Benutzer am System anmelden können, konzipieren Sie auch Testfälle, bei denen getestet werden soll, ob sich wider Erwarten auch nichtregistrierte Benutzer anmelden können.

## 4.9 Führen Sie Sicherheitstests nicht gleichzeitig mit fachlichen Tests durch

Ergebnisse von Sicherheitstests können ggf. destruktiv sein und möglicherweise die Software oder das System zum Absturz bringen. Daher sollten Sie solche Sicherheitstests in der Regel nicht parallel zu fachlichen Tests am gleichen zu testenden System durchführen.

## 4.10 Sehen Sie für alle ("normalen") Testfälle Negativbeispiele vor

Wenn Sie alle regulären funktionalen Tests durchführen, planen Sie für alle Negativbeispiele mit ein. Oft sind hier versteckte Schwachstellen zu finden, z.Bsp:

- Zugriffsversuch mit nichtberechtigtem User
- Upload einer virenverseuchten Datei (Achtung, nur mit EICAR-Teststring, sonst kann es evt. schiefgehen, siehe vorhergehende Regel)
- Authentisierung mit ungültigem Zertifikat
- Eingabe von „böartigem“ invalidem Input

## 5 Die 10 Goldenen Sicherheits-Regeln für „IT-Projektleiter“

Der Softwarelebenszyklus wird immer durch einen Projektleiter begleitet. Dieser muss den Überblick wahren und daher folgende Regeln beachten:

### 5.1 Sie sind verantwortlich für die Umsetzung aller Anforderungen, lesen Sie zunächst also alle anderen „Goldenen Regeln“ (You are responsible for the implementation of all requirements, first read all other „Golden Rules“)

Der IT-Projektleiter ist verantwortlich für alle Projektergebnisse und somit auch für die Umsetzung der Anforderungen des Auftraggebers. Dies beinhaltet natürlich auch die entsprechenden Sicherheitsanforderungen. Die Aufgabe als Schnittstelle zur IT besteht darin, die Anforderungen zu verstehen (ggf. gar einzufordern) und in konkrete Ergebnisse umzusetzen. Umgekehrt ist der Projektleiter auch verantwortlich für die zu planenden Ressourcen und insbesondere auch für die IT-Projektmitarbeiter, seien es schwerpunktmäßig Entwickler oder bei größeren Projekten auch spezielle Software-Architekten, Test-Teams etc..

Die erste „Goldene Regel“ für den Projektleiter besteht daher darin, sich zunächst einen Überblick über die Regeln aller anderen involvierten Zielgruppen bzw. Rollen zu verschaffen. So kann ein Verständnis der Sichtweisen und Aufgaben aller beteiligten Parteien erlangt werden und dies entsprechend in den Planungen und Maßnahmen frühzeitig und geeignet berücksichtigt werden.

### 5.2 Fordern Sie die fachlichen Sicherheitsanforderungen ein (Force the Business to clarify the Security Requirements)

Es ist allgemein anerkannt, dass eine der wichtigsten Aufgaben der Projektarbeit in der Auftragsklärung besteht. Hier gilt es die auftraggeberseitigen Erwartungen aufzunehmen und ein gemeinsames Verständnis zu schaffen. Aus Sicht eines Projektleiters gilt es, diese Anforderungen notfalls aktiv einzufordern, zu präzisieren und letztlich zu fixieren.

Dies gilt gleichermaßen auch für Sicherheitsanforderungen. Gerade bei Sicherheitsthemen besteht aber oft die stillschweigende Erwartungshaltung, dass es „halt sicher“ sein muss und „nichts passieren“ darf. Im Zweifel werden diese Anforderungen nicht näher diskutiert, so dass sie unausgesprochen bleiben und somit auch nicht präzisiert und fixiert werden. Um so größer ist später der Ärger, wenn sich dann im Nachhinein ergibt, dass dies in gewissen konkreten Konstellationen wider Erwarten doch nicht der Fall ist.

Diese stillschweigenden Erwartungen können aber oft unrealistisch hoch sein, Sicherheit kostet andererseits aber auch Geld und Aufwand. Eine wirkliche Gewährleistung einer Wiederanlaufzeit von maximal 4 Stunden setzt z.B. kostenspielige redundante Auslegung oder teure Onsite-Support-Verträge voraus, die Gewährleistung größtmöglicher Vertraulichkeit führt u.U. zu aufwändigen Authentisierungs-, Authorisierungs- oder gar Verschlüsselungsverfahren usw..

Auch bei dem Thema Sicherheit ist es also immer angeraten, die Erwartungen und Anforderungen ggf. aktiv einzufordern und zu fixieren, um Unklarheiten zu vermeiden. Ein pragmatischer Ansatz dazu ist in Regel 1 (Rolle Auftraggeber) skizziert. Bei kritischen Systemen ist eine fundierte IT-Risikoanalyse angeraten, die jedoch unter Hinzuziehung entsprechender IT-Sicherheits-Expertise durchgeführt werden sollte.

### 5.3 IT-Sicherheit gleich von Beginn an, nicht erst am Ende oder hinterher (IT-Security already at the beginning, not only at the end or afterwards)

Gerade beim Thema IT-Sicherheit ist manchmal der Ansatz zu beobachten, erst einmal die Anwendung zu erstellen, um dann – wenn überhaupt - am Ende noch einmal ein paar unspezifische Sicherheits-Checks nachzuschieben. Noch schlimmer ist nur noch die Tendenz, Sicherheitsprobleme im Nachgang erst dann erstmals zu betrachten, wenn im realen Betrieb schon ein Schaden eingetreten ist oder bemerkt wurde. Abgesehen davon, dass dann schon Schäden eingetreten sein können, ist eine solche nachträgliche oder gar „post mortem“ Behebung von Sicherheitsproblemen in der Regel viel aufwändiger als eine realistische und frühzeitige Berücksichtigung der Sicherheitsanforderungen.

Insofern gilt auch und vor allem bei IT-Sicherheit: Themen, Fragestellungen und Anforderungen der IT-Sicherheit sollten so früh wie möglich in einem Projekt Eingang finden. Nur aus bereits zu Projektbeginn definierten Sicherheitsanforderungen lassen sich die geeigneten und erforderlichen Sicherheitsmaßnahmen ableiten und entsprechend im weiteren Projektverlauf berücksichtigen.

### 5.4 In allen Projektphasen gibt es Arbeitspakete für IT-Sicherheit (Consider IT-Security in all project phases)

Sind entsprechend Regel 5.3 bereits zu Projektbeginn IT-Sicherheitsanforderungen definiert worden, gibt es analog zum gewohnten Vorgehen bei der Umsetzung fachlicher Anforderungen in einem Projekt in allen weiteren Phasen auch Arbeitspakete für IT-Sicherheitsthemen. Beinhaltet das Projektvorgehen beispielsweise Phasen für Fachkonzept – Design – Implementierung – Test/Review – Übergabe/Betrieb, so gestalten sich analog die entsprechenden Arbeitspakete für IT-Sicherheit:

- Im Fachkonzept werden die Anforderungen an die IT-Sicherheit geklärt und definiert.
- Im Design werden die Sicherheitsmaßnahmen entworfen, beschrieben und deren Deckungsgrad bzgl. der Anforderungen betrachtet.
- Bei der Implementierung werden die Sicherheitsmaßnahmen konkret umgesetzt.
- Bei Tests werden die Sicherheitsmaßnahmen hinsichtlich ihrer Wirksamkeit getestet.
- Auch nach Übergabe bzw. im Betrieb sollen die Sicherheitsmaßnahmen weiterhin vorhanden sein und auch wirksam und dauerhaft „gelebt“ werden können.

In den Arbeitspaketen bzgl. IT-Sicherheit können aber spezielle Maßnahmen, wie z.B. Code Reviews anfallen, spezielle Instrumente wie z.B. Sicherheits-Tools sinnvoll sein, oder sich spezielle Anforderungen wie etwa eine separate Testumgebung usw. ergeben.

Bzgl. der Sicherheitsthemen gibt es zwei sich ergänzende Ausprägungen:

Einerseits kann es analog zur „herkömmlichen“ Softwareentwicklung eigenständige Sicherheitsmodule geben, z.B. Berechtigungskonzepte, Verschlüsselungskomponenten usw., die den gleichen Erstellungszyklus wie „herkömmliche“ Softwarepakete durchlaufen.

Andererseits lässt sich IT-Sicherheit zusätzlich ganz allgemein auch als ein Qualitätsmerkmal der zu erstellenden Software auffassen. Insofern beinhaltet jede erstellte Software ein Stück Sicherheit. Eine Anforderung ist, dass sich die Software unter allen erwarteten Einsatzumständen (auch unter angenommenen Bedrohungsszenarien) wie erwünscht verhält und nicht etwa bei nicht bedachten Betriebszuständen (z.B. der Eingabe unvorhergesehener Daten) abstürzt oder vertrauliche Daten preisgibt.

## 5.5 Planen Sie hinreichende Ressourcen für IT-Sicherheit (Plan sufficient resources for IT-Security)

Auch IT-Sicherheit gibt es nicht für umsonst. Für IT-Sicherheit sind – wie für andere Aspekte der Softwareentwicklung - gleichermaßen Aufwände und Ressourcen vorzusehen. Wie groß der Anteil sein sollte, hängt natürlich wesentlich von den fachlichen IT-Sicherheitsanforderungen bzw. der Kritikalität der Anwendung ab.

Eine gute allgemeine Faustregel bei „ganz normalen“ Projekten ist ca. 5% des personellen Gesamtaufwandes. Bei öffentlich zugänglichen Internet-Anwendungen, bei denen z.B. immer mit Attacken und unzuverlässigem Input gerechnet werden muss, sollte dieser Anteil möglichst nicht unter 10% liegen.

Wie in Regel 5.4 erläutert, verteilt sich der Aufwand auf alle Projektphasen und muss nicht notwendigerweise aus eigenständigen Sicherheitspaketen bestehen, sondern kann ganz allgemein auch Qualitätssicherung aus dem Blickwinkel IT-Sicherheit beinhalten. Je nach Kritikalität müssen sich auch nicht unbedingt immer dedizierte Personen ausschließlich dem IT-Sicherheit widmen, ggf. kann auch die entsprechende zeitliche Berücksichtigung von IT-Sicherheitsaspekten hinreichend sein (vgl. auch Regel 5.6)

Bei sehr sicherheitskritischen Anwendungen sollte bereits in der Planungsphase spezielle IT-Sicherheitsexpertise hinzugezogen werden, die dann auch die geeignete Aufwandsschätzung vornehmen kann. Ggf. kann sich daraus dann auch die Notwendigkeit eines eigenständigen Teilprojektes IT-Sicherheit ableiten.

## 5.6 Vergessen Sie nicht Ihre (Projekt-)Mitarbeiter, planen Sie Sensibilisierung und Qualifizierung (Dont forget your employees, plan awareness and knowledge)

Die Qualifikation und Erfahrung der Projektmitarbeiter ist bekanntermaßen eine der wichtigsten Ressourcen bei der Entwicklung von Software. Dies gilt natürlich auch für das Thema IT-Sicherheit.

„Herkömmliche“ Software-Entwickler haben oft keine spezielle Kenntnisse oder Erfahrungen im Thema IT-Sicherheit, sondern sind vielmehr zunächst auf die fachliche Funktionsfähigkeit der von ihnen erstellten Module fokussiert. Entwickler sind sich dabei oft nicht einmal bewusst, wie einfach ggf. in der Software vorhandene Prüfungen ausgehebelt werden können (vgl. z.B. die Zehn Golden Regeln für Entwickler).

Die schlechte Alternative besteht darin, zunächst unbefangen die Software zu erstellen, um – wenn überhaupt – am Projekteende durch Sicherheitstests festzustellen, dass unvorgesehener Daten-Input oder andere Manipulationen wesentliche Sicherheitslücken aufzeigen.

Wesentlich zielführender ist es, bereits zu Projektanfang alle Beteiligten für die Belange der IT-Sicherheit zu sensibilisieren und Regeln an die Hand zu geben, wie man es richtig bzw. besser machen kann. Bzgl. der Entwickler stellen die „Zehn Goldenen Regeln für Entwickler“ einen guten ersten Einstieg für Sensibilisierung und Know-How-Aufbau dar.

## 5.7 Nutzen Sie Standards, sichere Komponenten und existierende Erfahrungen (Use and Reuse Standards, Trusted Components and existing Experience)

Grundsätzlich gilt die Devise: Aus Erfahrungen sollte man profitieren und nicht alle Fehler müssen wieder neu gemacht werden.

Dies gilt vor allem auch für das Thema IT-Sicherheit: Die jahrelange Erfahrung, die in gewisse spezielle Sicherheitsfunktionen oder ausgereifte Produkte eingeflossen ist, lässt sich nicht en passant in einem begrenzten Projekt neu erfinden. Daher gilt:

- Es macht keinen Sinn Schwachstellen selbst neu zu erfinden.
- Entwickeln Sie möglichst keine eigenen komplexen Sicherheitsfunktionen
- Setzen Sie – soweit möglich – auf bestehende, etablierte und vertrauenswürdige Lösungen

Gerade im Bereich IT-Sicherheit ist oft ein bestimmtes Problem schon irgendwo bewältigt worden. Für bestimmte benötigte Komponenten gibt es u.U. bereits Lösungen oder bestimmte Daten müssen gar nicht in eigenen Datentöpfen umständlich abgesichert werden, sondern können als zusätzliche Felder in einer etablierten und bereits gut abgesicherten Datenbank verwaltet werden.

Quellen für Erfahrungen und gute Lösungen kann es in verschiedensten Facetten geben:

- Bei eigenen Projektmitarbeitern, Teamkollegen, anderen Mitarbeitern
- Im eigenen Unternehmen, bei Vorgänger- oder Parallelprojekten, bei bereits bestehenden Anwendungen oder Infrastrukturen
- Bei Fremdentwicklung: In dem Unternehmen, für das die Anwendung erstellt werden soll
- Allgemein als Information, z.B. als Standard oder in einschlägiger Literatur
- Konkret als Lösung, z.B. schon vorhandene Grundfunktionen, als Kaufkomponente oder sogar frei verfügbares Modul

## 5.8 Bewahren Sie Handlungsspielraum für mögliche Veränderungen (Have Options for possible changes)

Unter dieser Regel lassen sich verschiedene mögliche Veränderungen (während des Projekts, vor allem aber auch nach Projektabschluss) zusammenfassen. Wichtige Regel ist dabei, dass Sie für sich, aber indirekt aber auch für den Auftraggeber hinreichend Handlungsspielraum bewahren, so dass möglichst flexibel und mit möglichst geringem Aufwand auf solche Veränderungen reagiert werden kann.

In der Praxis kann es natürlich auch während eines Projektes zu Änderungen der Sicherheitsanforderungen kommen, wobei bei hinreichend klarer anfänglicher Definition der Sicherheitsziele auch dem Auftraggeber klar sein dürfte, dass dies Mehraufwände nach sich ziehen kann. Gegen Projektende, z.B. in der Testphase oder bei anschließenden speziellen Sicherheitstests können sich Fehler (auch bei Sicherheitseigenschaften) herausstellen, die es zu beheben gilt und die Sie sich selbst zurechnen müssen. Bei entsprechend modularem Aufbau ist es immer einfacher, solche Änderungen zu berücksichtigen bzw. Fehler korrigieren zu können.

Insbesondere können sich aber auch später Veränderungen ergeben, z.B. nachträglich festgestellte Sicherheitsmängel, die es z.B. im Rahmen von Gewährleistung oder Wartung zu beheben gilt. Gleichmaßen können nach Auslieferungen neue Bedrohungsszenarien auftreten, die bislang noch nicht bedacht oder gänzlich unbekannt waren.

Typischerweise wird die Behebung solcher Sicherheitslücken in Form von „Security Patches“ erstellt und dem Kunden oder Auftraggeber zur Verfügung gestellt. Der Zeitfaktor kann dabei oft kritisch sein, da die Sicherheitslücke u.U. schon bekannt ist und es zu befürchten ist, dass die Lücken auch real ausgenutzt werden. Umgekehrt muss aber auch damit gerechnet werden, dass die Funktionsfähigkeit der Software für den Kunden eine ganz wesentliche Rolle spielt.

Die Wartbarkeit und Pflegbarkeit spielt also auch und insbesondere in Hinblick auf IT-Sicherheit eine ganz wesentliche Rolle, und es muss gewährleistet sein, dass z.B.:

- Der aktuelle Security Patch-Level jederzeit einfach festgestellt werden kann
- Security Patches möglichst nutzerfreundlich eingespielt werden können
- Es immer auch Rollback-Möglichkeiten gibt

Allgemeine Grundlage und Voraussetzung ist ferner eine etablierte und vertrauenswürdige Kommunikationsstrategie für den Umgang mit möglichen Sicherheitsproblemen und sich daraus ergebendem Handlungsbedarf.

## **5.9 Machen Sie Restrisiken transparent und lassen sich diese abzeichnen (Achieve Awareness about Residual Risks and let them „Sign Off“)**

Absolut 100%ige Sicherheit ist grundsätzlich nicht möglich und wäre wirtschaftlich auch kaum sinnvoll. Dies lässt sich exemplarisch beispielhaft am Thema „Verfügbarkeit“ erläutern: Garantierte 99%ige Verfügbarkeit im Sinne 7d24h ist schon nur mit sehr großem Aufwand erreichbar, 99,9%ige Verfügbarkeit kaum darstellbar und finanzierbar, im Prinzip steigt der Aufwand exponentiell und 100% ist grundsätzlich nicht erreichbar. Es muss also ein vernünftiger und wirtschaftlicher Kompromiss zwischen Verfügbarkeit und dafür vorzusehendem Aufwand gefunden werden.

Dies gilt analog für andere Sicherheitsziele wie Vertraulichkeit, Integrität usw. und z.B. trotz aller technischen Maßnahmen für z.B. Vertraulichkeit wird es immer Personen geben, die im Endeffekt auf die Daten zugreifen können und zur Ausübung Ihrer Tätigkeit auch müssen.

Insofern gibt es immer auch Restrisiken, die bewusst in Kauf genommen werden müssen. Für einen Projektleiter ist es empfehlenswert, zumindest die wesentlichen verbliebenen Restrisiken zu dokumentieren und so dem Auftraggeber transparent zu machen.

Darüber hinaus gibt es natürlich auch allgemeine Risiken, die später im laufenden Betrieb eintreten können. Beispielsweise kann es sein (bzw. ist sogar sehr wahrscheinlich), dass sich bei gewissen Kaufkomponenten (z.B. dem mitbenutzen Standard-Webserver) später eine Sicherheitslücke herausstellt, für die der Hersteller dann auch einen Sicherheits-Update herausgibt der zeitnah eingespielt werden sollte. Auch wenn der Projektleiter dies möglicherweise gar nicht selbst beeinflussen kann, so ist es für den Auftraggeber sicherlich hilfreich, wenn die wichtigsten solcher Restrisiken benannt sind, und die entsprechenden Folgemaßnahmen beschrieben sind.

## **5.10 Denken Sie an diejenigen, die später mit dem Produkt umgehen müssen (Consider those people, who later have to work with the results)**

Irgendwann ist das Projekt zu Ende, alle Tests sind erfolgreich absolviert und das Ergebnis soll vom Auftraggeber abgenommen werden und in den laufenden Betrieb übergeben werden. Im besten Falle sind gemäß den oben geschilderten Regeln im Projektverlauf auch Belange der IT-Sicherheit angemessen berücksichtigt worden.

Nun treten jedoch neue Gruppen auf den Plan, nämlich diejenigen, die im täglichen Geschäft mit der erstellten Anwendung bzw. dem Produkt umgehen müssen. Dabei kann es sich um Administratoren handeln, die die Software installieren, betreiben und warten sollen. Vor allem gibt es aber natürlich auch diejenigen,

die die Anwendung im täglichen Geschäft nutzen sollen. Dabei kann es sich je nach Anwendungszweck um interne Mitarbeiter, externe Endkunden oder weitere Nutzerkreise handeln.

Während es oft naheliegend erscheint, dass schon im Projektverlauf gewisse Anleitungen und Hinweise zur funktionalen Bedienung des Produktes erstellt werden müssen, werden entsprechende Erläuterungen bzgl. des Themas IT-Sicherheit gerne vergessen. Dabei hängt aber auch die Sicherheit im späteren Betrieb wesentlich davon ab, daß die Nutzer wissen, wie sie mit dem Produkt umzugehen haben und worauf sie achten müssen. Insofern ist darauf zu achten, dass die in der Regel ohnehin zu erstellenden Anleitungen und Hinweise auch entsprechenden Hinweise zum Umgang mit der IT-Sicherheit beinhalten. Beispiele dafür sind

- Beachtung gewisser Merkmale (z.B. https und „Schlößchen“ in der Browserleiste)
- Gewisse allgemeine Verhaltensregeln (z.B. keine Passwortweitergabe)
- Spezielle Verhaltensregeln (z.B. Reaktion bei gewissen Fehlersituationen)

Ein wichtiger Aspekt besteht dabei auch darin, den sicheren Umgang mit dem Produkt möglichst einfach und verständlich zu gestalten. Je stringenter und verständlicher die entsprechenden Regeln und Prozesse sind, desto weniger muss umständlich in dicken Handbüchern erklärt werden, die im Zweifel in der Realität ja leider ohnehin kaum komplett gelesen werden.