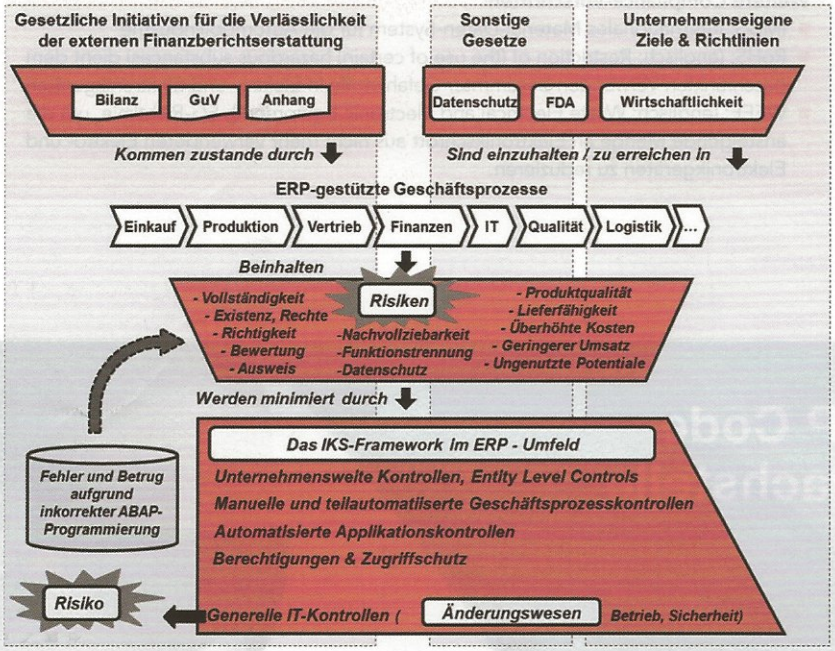
Compliance und Applikationssicherheit im ERP-Umfeld

MIT SCHWACHSTELLEN UMGEHEN UND SIE UNTER KONTROLLE HALTEN

Applikationssicherheit wird von vielen SAP-Nutzern falsch eingeschätzt. Aussagen wie „Unser IKS ist von Wirtschaftsprüfern als effektiv eingestuft worden" oder „Unsere Entwickler wissen, was sie tun" überschatten den meist akuten Handlungsbedarf. Unternehmen müssen ein Verständnis für die Compliance-Risiken entwickeln, die durch unsicheren ABAP Code entstehen.



Risiken im Bereich Sicherheit & Programmierung aus IKS- und Compliance-Sicht.

Von Maxim Chuprunov, RISCOMP & Andreas Wiegenstein, Virtual Forge

Das Stichwort „Compliance" hat über die letzten Jahre hinweg Einzug in den Alltag von CIOs und CFOs gehalten: Wirtschaftskrisen, Unternehmensskandale, zunehmender Wettbewerbsdruck und immer raffiniertere IT-Kriminalität haben dafür gesorgt, dass die komplexe Welt der ERP-gestützten Geschäftsprozesse oft nicht minder komplexen Compliance-Anforderungen genügen soll. Als Folge wurden Rufe nach effizienten IKS (Internes Kontrollsystem)-Prozessen immer lauter. Inzwischen verfügen viele Unternehmen über vorzeigbare Prozesse für IKS und Compliance Management, die alle relevanten Bereiche abdecken sollen. Abbildung 1 zeigt, dass diverse Risiken die Compliance-Ziele in ERP-gestützten Geschäftsprozessen gefährden können.

Wie ein IKS den Risiken entgegenwirkt, ist zum Beispiel in den international anerkannten Referenzmodellen wie COSO & COBIT beschrieben. In einer üblichen IKS-Struktur sind die sogenannten Generellen IT Kontrollen (ITGC – IT General Controls) die Voraussetzung für das Erreichen sämtlicher IKS-Ziele in einem IT-dominierten Umfeld. Die ITGC müssen sehr verlässlich sein, da sie im Fokus der internen und externen Revision stehen. Doch dieses Fundament hat in der Praxis oft Schwachstellen. Aus IKS-Sicht weist insbesondere das Änderungswesen (Change Management) bei Eigenentwicklungen starkes Verbesserungspotential auf.

Ein falsches Sicherheitsgefühl

Immer mehr Unternehmen führen moderne GRC-Software (Governance, Risk & Compliance) ein. Mehrwert bringen das Automatisieren der IKS-Prozesse sowie die Möglichkeit, diese mit relevanten Abläufen in Risk Management, Benutzer- und Berechtigungsverwaltung, Richtlinienverwaltung et cetera fest zu verdrahten. Allein die Integration von GRC-Anwendungen mit ERP-Systemen ist sehr vielversprechend: Stamm-, Bewegungs- und Steuerungsdaten in ERP-Systemen werden effizient überwacht, die Risiken und Schwachstellen werden dabei auf Knopfdruck erkannt und Behebungsmaßnahmen werden automatisch eingeleitet. Dieser Continuous Control Monitoring (CCM) genannte Ansatz stößt allerdings bei klassischer GRC-Software an technische Grenzen, wenn es um das Prüfen von ABAP-Code geht.

Bei manuell getriebenen IKS-Prozessen sieht es kaum anders aus: Zum einen ist manuelles Prüfen mit hohem Aufwand verbunden – selbst wenn es nur aus Stichproben besteht. Zum anderen haben Prüfer üblicherweise auch keine Werkzeuge zum vollständigen Prüfen. Zumal solche Prüfungen sehr spezifisches Fachwissen erfordern. Ob IKS-Prozesse Excel-basiert ablaufen oder durch GRC-Software unterstützt werden – der produktive ABAP-Code bleibt eine Black Box.

Welche Gefahren gehen von Sicherheitslücken im ABAP-Code aus?
Die Risiken durch unsichere Programmierung lassen sich in sechs Bereiche einteilen. Sie beschreiben jeweils die Auswirkung von Sicherheitsdefekten auf die Compliance und welche gesetzlichen Anforderungen verletzt werden, wenn unsicherer Code auf ein Produktivsystem gelangt.

Risiko 1 – Unberechtigtes Ausführen von Business Logik
Fragestellungen bezüglich der Zugriffsberechtigungen sind in den meisten Compliance-Anforderungen enthalten, die sich direkt oder indirekt auf Daten und Verarbeitungslogik in IT-Systemen beziehen:

- Richtigkeit und Zuverlässigkeit der Finanzberichtserstattung (SOX, GobS et cetera)
- Qualität (FDA)
- Datenschutz / Data Privacy
- Steuerrecht

Das Einhalten der Prinzipien der minimal notwendigen Rechtevergabe sowie Funktionstrennung kann in diesen Domänen beeinträchtigt werden. Strafrecht kommt unter Umständen ebenfalls zum Tragen.

Risiko 2 – Unberechtigtes Auslesen von Geschäfts- und Konfigurationsdaten
Unberechtigter Lesezugriff ist in erster Linie aus der Perspektive der Datenschutzanforderungen (insbesondere HR-Daten) sowie Anforderungen an den Schutz von Kreditkartendaten relevant. Diebstahl von geschäftskritischen Daten wie dem Kundenstammkann dem Strafrecht unterliegen.

Risiko 3 – Unberechtigtes Verändern von Geschäfts- und Konfigurationsdaten
Direktes Ändern der Bewegungsdaten kann zum Nichteinhalten des Belegprinzips führen, insbesondere für die Finanzberichtserstattung. Es ist direkter Verstoß gegen elektronisches Radierverbot § 239 HGB und ist auch hinsichtlich qualitätsbezogener gesetzlicher Anforderungen relevant (DA, EU-Richtlinien für GMP, GLP et cetera). Strafrecht kann bei bestimmten Voraussetzungen ebenfalls relevant sein

Risiko 4 – Gefährden der Verfügbarkeit des Systems
Bei kritischen Systemen sind die Compliance-Anforderungen im Bereich der Finanzberichtserstattung relevant, weil ein Wirtschaftsprüfer im Interesse der Anleger beim Erkennen der potentiellen Gefährdung des Fortbestands des Unternehmens entsprechende Feststellungen ordnungsgemäß kommunizieren muss. Bei absichtlicher Gefährdung kann Strafrecht greifen.

Risiko 5 – Beeinträchtigung der Nachvollziehbarkeit von Geschäftsvorgängen
Gesetzliche Anforderungen wie der § 239 HGB und andere Vorschriften verlangen die Nachvollziehbarkeit von Entstehen und Ändern der im System aufgezeichneten Geschäftsvorfälle (Belegprinzip) sowie der Stamm- und Steuerungsdaten.

Risiko 6 – Identitätsdiebstahl
Der Grundsatz der Accountability und der Nachvollziehbarkeit kann beeinträchtigt werden (Finanzberichtserstattung, Qualität, Datenschutz). Identitätsdiebstahl mit dem Ziel, zusätzliche Berechtigungen zu erlangen, zieht die gleiche Compliance-Relevanz nach sich wie ein Risiko 1 beschrieben.
Der Missbrauch der meisten (Sicherheits)Lücken setzt böse Absicht, kriminelle Energie und häufig auch Innentäter voraus. Das senkt zwar die Eintrittswahrscheinlichkeit, aber aufgrund des großen Schadenspotentials sind entsprechende Risiken durchaus ernst zu nehmen.
Diese Risiken können sich verschärfen, wenn die Entwicklung an Beratungshäuser ausgelagert wurde. Hier ist dann nicht mehr ersichtlich, welches Know-How die Entwickler haben oder wie loyal sie gegenüber dem Auftraggeber sind. Auch Unternehmenszukäufe erhöhen das Risiko, da Unmengen an neuem ABAP-Code ins Unternehmen fließen können. Entwickler in Tochtergesellschaften sollten nicht mittels Hintertüren im Code auf Daten der Muttergesellschaft zugreifen können.

Beispiele von Sicherheitsdefekten im ABAP Code

Es gibt viele verschiedene Arten von Schwachstellen, die durch ABAP-Code entstehen können. Um beispielhaft auf

ID	Schwachstelle	Beschreibung	Risiken
APP-01	ABAP Command Injection	Ausführung von beliebigem ABAP Code	1-6
APP-02	OS Command Injection	Ausführung beliebiger Betriebs system-Kommandos	1, 4
APP-03	Improper Authorization (Missing, Broken, Proprietary, Generic)	Fehlende oder fehlerhafte Berechtigungsprüfung	1
APP-04	Generic Module Execution	Unerlaubte Ausführung von Modulen (Reports, FuBa's, etc)	1
APP-05	Cross-Client Database Access	Mandantenübergreifender Zugriff auf Geschäftsdaten	2, 3
APP-06	SQL Injection	Schadhafte Manipulation von Datenbankbefehlen	2, 3, 4, 5
APP-07	Unmanaged SQL	Verwendung nativer Datenbank befehle	4, 5
APP-08	Cross-Site Scripting	Manipulation des Browser UI, Diebstahl von Berechtigungen	6
APP-09	Cross-Site Request Forgery	Ausführung von Business Logik im Namen eines anderen Benutzers	6
APP-10	File Upload (Malware)	Speicherung schadhafter Dateien auf dem SAP Server	1, 4
APP-11	Directory Traversal	Unerlaubter Schreib-/Lesezugriff auf Dateien (SAP Server)	2, 3, 4

die technischen Risiken eingehen zu können, soll hier nur die BIZEC (The Business Security Initiative) APP/11 Liste als Basis verwendet werden.

Das anschaulichste Sicherheitsproblem im ABAP-Code sind fehlende oder fehlerhafte Berechtigungsprüfungen (APP-03). Da ABAP ein explizites Berechtigungskonzept hat, muss jedes Programm, das berechtigungsrelevante Aktionen ausführt, auch die zugehörigen Berechtigungsobjekte selbst überprüfen. Fehlt diese Prüfung im ABAP oder ist sie fehlerhaft, dann führt das Programm die Aktion aus, auch wenn der Benutzer die eigentlich erforderliche Berechtigung gar nicht hat.

Ein zweites grundsätzliches, aber weniger offensichtliches Problem sind sogenannte Injection-Schwachstellen. Durch diese können bösartige Benutzer bestimmte ABAP-Befehle zur Laufzeit schadhaft verändern. Durch Manipulation dynamischer ABAP-Befehle kann sogar trotz korrekter Berechtigungsprüfung manipuliert werden. Häufig wissen ABAP-Entwickler nicht, welche Risiken beim Einsatz dynamischer Programmiertechniken entstehen und öffnen damit ungewollt Hackern Tür und Tor.

Beispiel einer ABAP Command Injection (APP-01) Schwachstelle:

```
REPORT Z_RISIKO.

DATA lt_prog(72) OCCURS 0 WITH
HEADER LINE.
PARAMETERS lv_name TYPE string.

lt_prog = 'REPORT ZFT.'.
APPEND lt_prog.

CONCATENATE
  `DATA lv_tmp(80) TYPE c VALUE '`
  lv_name
  `'.`
  INTO lt_prog.
APPEND lt_prog.

lt_prog = 'WRITE / lv_tmp.'.
APPEND lt_prog.

INSERT REPORT 'ZFT' FROM lt_prog.

SUBMIT ('ZFT').
```

Das Code-Beispiel zeigt, wie Benutzerinput (lv_name) zur Laufzeit verwendet wird, um einen neuen Report zu erstellen, der dann auch gleich ausgeführt wird. Ein Angreifer kann das Programm so manipulieren und ohne Entwicklerberechtigung auf einem Produktivsystem Code schreiben.

Mögliche Wege zur Prävention und zum Aufdecken von Fehlern im Code

Die besten Kontrollmechanismen aus IKS- und Compliance-Sicht sind vorbeugender Natur. Für generelle IT-Kontrollen (ITGC) im Bereich Änderungsmanagement bei Eigenentwicklungen sind in der Praxis folgende Mechanismen zu empfehlen:

- Ordnungsgemäße Entwicklerrichtlinien & Standards
- Einsatz von statischen Code Analyse Tools

Standards: Ohne Entwicklungsstandards ist ein sicherheitsbewusstes Programmieren nicht möglich. Alle Entwickler und Zulieferer müssen diese Entwicklungsstandards einhalten. Wichtig: Das Beheben von Sicherheitsfehlern im Code kann deutlich aufwändiger sein, als beispielsweise Änderungen an Konfiguration und Benutzer-Rollen. Je mehr Code ein Unternehmen geschrieben hat, desto größer ist das potentielle Risiko. Es empfiehlt sich daher frühzeitig, die Entwicklungsstandards in Punkto Sicherheit ordnungsgemäß zu gestalten, damit die Menge an Problemen durch neuen Code nicht stetig steigt.

Qualitätssicherungstools: Angemessene Qualitätssicherungsprozesse sind ein Muss. Damit Schiefstände und Differenzen zum Soll-Zustand im Vorfeld aufgedeckt werden, können Tools zur statischen Code-Analyse wie Virtual Forge CodeProfiler in den Korrektur- und Transportwesen-Prozess von SAP integriert werden. Somit wird verhindert, dass unsicherer Code in die Produktion gelangt.

Mit diesen Tools lässt sich natürlich auch der in der Produktion befindliche ABAP-Code nachträglich scannen – und so das immense Compliance-Risiko eindämmen, dass von ABAP-Code ausgeht.

(ur) @