Scheunentore in SAP-Systemen und wie man sie schließt

Zunehmend löchrig

Ralf Wenzel, Daniel Lehmann

Über gehackte SAP-Systeme dringt selten etwas an die Öffentlichkeit. Denn kein Unternehmen spricht gern darüber, dass seine Anwendungslandschaft – meist randvoll mit Eigenentwicklungen und intern gewartet –, Angriffen von innen oder außen nicht standhält. Das wäre deutlich peinlicher als Sicherheitslücken in gekaufter Closed-Source-Software. Gleichwohl sind solche Fälle allgegenwärtig.

In der Vergangenheit ließ sich die Sicherheit in einer SAP-Umgebung einfach gewährleisten: Zugriff nur über das hauseigene Netzwerk mit dem proprietären SAP GUI – Schnittstellen gab es nur zu vertrauenswürdigen Systemen (Abbildung). Man musste lediglich alle Berechtigungen ordentlich pflegen und in den Eigenentwicklungen sorgfältig prüfen. Den Rest erledigten die Walldorfer. Und genau daraus resultieren die heutigen Schwierigkeiten.

Denn inzwischen erlauben Shop-Anwendungen den mittelbaren Zugriff auf SAP-Systeme aus dem Internet heraus. Web Dynpro und Business Server Pages drängen das SAP GUI als Benutzerschnittstelle an den Rand, und das SAPUI5 SDK, ein nagelneues Framework zum Entwickeln von Webanwendungen, das komplett auf HTML5 und JavaScript setzt, soll es gar ablösen. Hier entstehen Sicherheitslücken, mit denen erfahrene SAP-Experten nicht gut umgehen können.

Aber nicht nur das Neue schafft Probleme, sondern auch klassische und fast vergessene Techniken. Alte Hasen mögen es kaum glauben, aber in einer Unzahl von SAP-Installationen entspricht das Kennwort des CPI-C-Benutzers (Common Programming Interface – Communications) immer noch dem bei der Einrichtung gesetzten Default. Viele Betreiber ändern es nicht, weil das Thema CPI-C aus ihrer Sicht schon lange nicht mehr relevant erscheint. Weiterhin seien Interessierte auf den Report RS_REPAIR_SOURCE hingewiesen, den man bequem mit der Transaktion *SA38* ausführen kann, um Code ins System einzuschleusen. Erste Kommentarzeile: „Use it very, very, very carefully!“ Ebenso lesenswert: Die Reports RC1TCG3Y und RC1TCG3Z, die beliebige Daten vom SAP-Server zum Frontend kopieren (und umgekehrt). Hier gilt es, die Hinweise 1552798 und 1505368 zu beachten.

SAP-Experten müssen umlernen

Der Anschluss von SAP an die Außenwelt zwingt einige Berufsgruppen dazu, ihre Hausaufgaben in weiten Teilen neu zu machen. Insbesondere betrifft das die IT-Leitung, die Administration, die Verantwortlichen für das Berechtigungskonzept und – das wird gern vergessen – die Entwickler.

Unter der Hand zeigt sich ein steigendes Interesse an Sicherheitschecks, Schulungen und Audits. Das merkt ein Berater besonders, wenn er offenbart, dass er auch in Sachen Datenschutz berät. Nicht etwa, weil der Kunde durch NSA und Abhöraffären ein gesteigertes Sicherheitsbewusstsein entwickelt hat, sondern schlichtweg aufgrund einer zunehmenden Zahl von Angriffen auf das hauseigene System. Sie kommen mal von innen, immer öfter jedoch von

außen, was jeweils unterschiedliche Sicherheitsstrategien verlangt.

Auch die Politik gelangt langsam zur Erkenntnis, dass sich neue Technologien nicht mit alten Gesetzen regeln lassen, und produziert permanent neue Bestimmungen. Big Data und HANA-Datenbanken verwischen die technischen Grenzen bei der Analyse riesiger Datenbestände, und die Anforderungen an Compliance und Transparenz wachsen, etwa durch den Sarbanes-Oxley Act (SOX). Leider ist das Bundesdatenschutzgesetz das wohl am wenigsten eingehaltene Gesetz nach der Straßenverkehrsordnung. Viele Unternehmen sehen darin allein einen Kostenfaktor, dabei würde ihnen praktizierter Datenschutz enorm helfen. Beim geplanten IT-Sicherheitsgesetz, das kritische Infrastrukturen vor Hacker-Angriffen schützen soll, ist die Sache offensichtlicher. Es besteht die Hoffnung, dass die Betroffenen künftig gewissenhafter reagieren (siehe „Alle Links“).

Was der Gesetzgeber nicht regelt, regeln die Banken, denn sie haben inzwischen gemerkt, dass IT-Sicherheit und Datenschutz existenzielle Unternehmenswerte bewahren können. Außerdem verfügen sie über genügend Einfluss, um entsprechende Richtlinien (BASEL I, II, III) durchzusetzen.

Da Geschäftsführungen verpflichtet sind, die Unternehmenswerte vor Verlust oder Wertminderung zu schützen, müssen sie mit ihren Rechtsabteilungen und IT-Leitungen auf Augenhöhe kommunizieren. Kompetenzgerangel bei der IT-Sicherheit ist vorprogrammiert: Systemverantwortliche gegen Revisoren gegen Modulverantwortliche gegen Entwickler. Und im Datenschutzrecht kommt noch das fehlende Konzernprivileg hinzu. Maßnahmen, die der Konzern einleitet, für die aber die untergeordneten Firmen haften müssen, führen oft zum unternehmensübergreifenden Streit.

Ohne korrekte Analyse läuft nichts

Administratoren und Systembetreuer sind die ersten Ansprechpartner, wenn es um IT-Sicherheit geht. Von entscheidender Bedeutung ist, sie von Anfang bis Ende in alle IT-Prozesse einzubinden. Sie sollen in erster Linie die Angriffsfläche, die Webservices, Gateways, J2EE et cetera bieten, möglichst klein halten und Security Notes rechtzeitig einspielen. Hierzu müssen die Verantwortlichen die datentechnischen Unternehmenswerte analysieren und bewerten, denn jedes Datum hat einen Wert in Euro und Cent, der unbedingt bekannt sein muss. Nur so lassen sich der Schutzbedarf klassifizieren und die Risiken abschätzen. Wichtig ist hierbei ein kontinuierliches, automatisiertes Monitoring.

Ein bisschen surreal mutet es an, in geschätzten Arbeitskollegen Subjekte zu sehen, die versehentlich oder vorsätzlich die IT destabilisieren können. Gleichwohl ist dieser Blickwinkel notwendig, um geeignete Sicherheitsmaßnahmen (zum Beispiel bei eigenen ABAP-Programmen) zu ermitteln und zu ergreifen.

Quelle: SAP

In den 80er-Jahren war es noch einfach, SAP-Systeme gegen die böse Außenwelt abzuschotten. Mit der Öffnung Richtung Internet änderte sich die Lage dramatisch.

Fehlbedienungen gehören dabei zu den einfacheren Fällen. Daraus kann aus Unachtsamkeit oder Leichtsinn ein Schaden entstehen. Beim Fehlverhalten sieht die Sache schon anders aus: Man muss in die Rolle des Advocatus Diaboli schlüpfen und sich ausmalen können, wie eine Person oder eine Gruppe vorgehen könnte, wenn sie sich zulasten des Unternehmens bereichern will (denken wie ein potenzieller Angreifer mit internem Wissen). Hier ergibt sich ebenfalls eine Schnittstelle zwischen IT und einer anderen Abteilung, der Revision.

Ein ordentliches und in sich schlüssiges Berechtigungskonzept zu planen und umzusetzen ist keine leichte Aufgabe. Es sollten speziell geschulte Mitarbeiter oder externe Berater zum Zuge kommen, die um die Tücken und hausgemachten Fallen des Berechtigungswesens wissen. Basis der zugehörigen Logik ist die sogenannte rollenbasierte Zugriffskontrolle (Role Based Access Control, RBAC), eine der TOMs (technischen und organisatorischen Maßnahmen) im Bundesdatenschutzgesetz. Daran lässt sich die Relevanz eines guten Berechtigungskonzepts ablesen: Es gehört zu den rechtlich notwendigen Maßnahmen, was oftmals in dieser Tragweite nicht bekannt ist.

Rollen richtig besetzen

Ein Berechtigungskonzept soll die gesamte betriebswirtschaftliche Struktur des Unternehmens ohne Konflikte der Funktionstrennung technisch abbilden. Es enthält unterschiedliche Organisationsbegriffe: Der institutionelle beschreibt, wo ein Mitarbeiter innerhalb der Organisation einsortiert wurde. Der instrumentelle legt fest, welche Aufgaben er hat. Da jeder Angestellte in der Regel mehrere erfüllen muss, bekommt er nicht nur eine Rolle zugewiesen. Dabei ist der sogenannte normative Berechtigungsbegriff zu beachten: Das Besetzen von Rollen muss rein organisatorisch erfolgen, also ohne Berücksichtigung einer konkreten Person. Kritische Berechtigungen sollten die Verantwortlichen restriktiv

iX-TRACT

- Erfolgreiche Angriffe auf SAP-Systeme gibt es häufig. Weil das den Unternehmen peinlich ist, dringt selten etwas darüber an die Öffentlichkeit.
- Früher war es relativ einfach, SAP-Systeme gegen die Außenwelt abzuschotten. Ihre Öffnung hin zum Internet offenbart nun jede Menge Sicherheitslücken.
- Alte SAP-Hasen müssen umdenken und umlernen – nur mit einem neuen Bewusstsein für die Gefahren lassen sich erfolgversprechende Sicherheitskonzepte in der Standardsoftware etablieren.

vergeben, sowohl im Produktivsystem als auch in Testumgebungen. Das ist allein schon deshalb notwendig, weil es sich bei Letzteren oft um Kopien des Produktivsystems handelt.

Eigentlich müsste jeder Datenschutzbeauftragte, jeder externe Datenschutzberater und jeder Revisor es strikt verbieten, dass Produktivsysteme auf Testsysteme mit deutlich gelockerten Berechtigungen kopiert werden. In der Regel spielt ein Administrator eine Sicherung des Produktivsystems auf einem Testsystem ein und erledigt ein paar Nacharbeiten, bei denen er zum Beispiel die System-ID in einigen Tabellen „gerade zieht“. Zwar ist solch eine Kopie ungemein praktisch bei Wartung und Entwicklung (insbesondere bei der Fehlersuche). Hier können Entwickler, Modulverantwortliche und Key-User unter realistischen Bedingungen Belege buchen, Formulare drucken und im System nach Fehlern suchen (inklusive der Berechtigung zum Verändern von Variableninhalten). Das grenzenlose Vertrauen in die (teilweise externen) Mitarbeiter lässt sich jedoch nur als bodenloser Leichtsinn bezeichnen. Im Zweifel folgen deutliche haftungsrechtliche Verwerfungen, wenn ein Anwender die gegebenen Möglichkeiten nutzt, um dem Unternehmen zu schaden. Jeder erfahrene SAP-Berater kennt Fälle, in denen Bestellbuchungen aus dem Testsystem versehentlich an Lieferanten versendet wurden, die für sie nicht als Testbestellungen erkennbar waren.

Auch wenn man nicht (versehentlich oder absichtlich) schreibend ins System eingreift, ist schon der erweiterte Lesezugriff etwa auf Preise, Rabatte und Lieferkonditionen, die vor denselben Benutzern im Produktivsystem bewusst abgeschirmt sind, kritisch. Hier verbietet es sich, den Anwender (in der Allgemeinheit seines Begriffes) als Freund des Unternehmens zu betrachten. Ein gesundes Misstrauen schadet nicht – zu viel Vertrauen schon. Denn schnell gelangt man bei der Frage nach dem Schuldigen eines unberechtigten Zugriffes zu dem Argument, dass die Daten ja auf dem Silbertablett serviert wurden.

Entwickler als Sicherheitsrisiko

Unternehmen, die ihre wertvollen Informationen nicht leichtsinnig preisgeben wollen, sollten Software wie SAPs Test Data Migration Server einsetzen, die alle betroffenen Daten inklusive Verknüpfungen und Bezüge pseudonymisieren. Zwar erschwert das die Arbeit des Wartungsteams und der Entwickler, muss aber sein. Denn die Situation ist ausgesprochen beängstigend: Den Autoren dieses Artikels ist bislang kein einziger Entwickler begegnet, der eine spezielle Schulung zum Datenschutz oder zur SAP-Systemsicherheit besucht hätte. Meist erhielten sie nicht einmal die nach Bundesdatenschutzgesetz für alle Mitarbeiter vorgeschriebenen Datenschutzunterweisungen. Das mag bei „normalen“ Angestellten als lässlicher Gesetzesverstoß durchgehen, bei IT-nahen Kollegen ist es eine nicht hinnehmbare Fahrlässigkeit, die in Rechtsstreitigkeiten dem Unternehmen angelastet wird. Jeder Entwickler, ob intern oder extern, hat weitreichende Möglichkeiten der Systemmanipulation und stellt im Falle von Implementierungsfehlern oder vorsätzlichem Fehlverhalten ein latentes Sicherheitsrisiko dar.

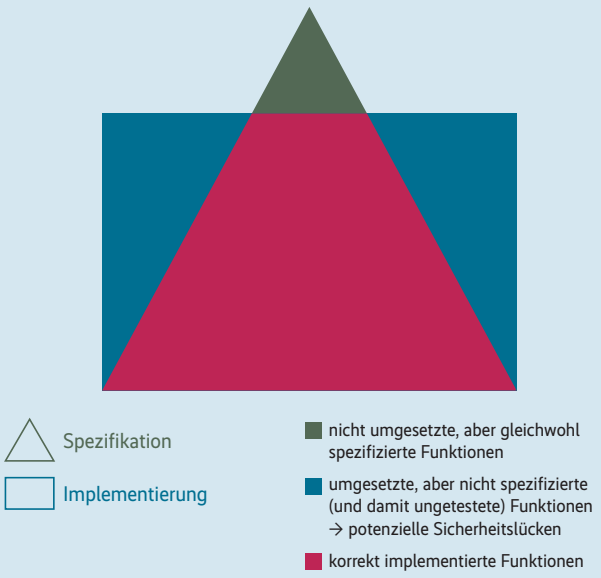
Das Erzeugen vermeintlicher Sicherheit durch das Beschränken von Benutzer-Accounts auf Nicht-Produktivsysteme kann als Negativbeispiel herhalten. Denn gewiefte Profis können ohne Weiteres über RFCs (Remote Function Calls) auf alle Systeme im Systemverbund zugreifen, zum Beispiel mit dem (inzwischen entschärften) Funktionsbaustein *EBA_TABLE_SELECT*. Zu dieser

Spezifikation vs. Implementierung

Gute Softwareentwicklung ruht auf den drei Pfeilern Vertraulichkeit, Integrität und Verfügbarkeit. Hört sich gut an, in der Praxis beginnen die Schwierigkeiten jedoch schon beim Formulieren der Anforderungen an eine Eigenentwicklung. Einige der Beteiligten legen fest, welche Funktionen der Anwender bekommen soll. Nicht selten passt so eine Anforderung auf einen Bierdeckel. Datenverarbeitungs-, Test- und Sicherheitskonzept: Fehlanzeige. Und die Dokumentation findet auf dem Bierfilz sowieso keinen Platz. Hier sparen die Unternehmen an der falschen Stelle.

Anforderer und Entwickler sprechen oft unterschiedliche Sprachen, was allein schon enorme Sicherheitsprobleme verursachen kann. Die Gefahr liegt allerdings abseits der Vereinigungsmenge von Spezifikation und Implementierung. Teile der Spezifikation, die es nicht in die Implementierung schaffen, fallen beim Test auf. Das Risiko besteht in dem Teil der Implementierung, der nicht spezifiziert wurde. Denn unerwünschte Funktionen, mit denen Anwender (und oft auch Entwickler) nicht rechnen, werden nicht getestet und stellen daher ein erhebliches und völlig unbekanntes Sicherheitsrisiko dar.

Professionelles Arbeiten sieht anders aus. Der erste Schritt muss eine Anforderung in Form eines vollständigen Fachkonzepts sein, das vor allem die Wechselwirkungen mit anderen Prozessen beschreibt. Im zweiten Schritt erstellt der Entwickler ein Datenverarbeitungskonzept, das die technische Umsetzungsplanung und eine belastbare Aufwandsschätzung enthält und das als Basis für die spätere Dokumentation dient. Nach Abschluss der Arbeiten werden Tests und Abnahme nach einem professionellen, strukturierten und vollständigen Testkonzept durchgeführt, und zwar nicht durch das Durchspielen beispielhafter Anwendungsfälle. Als Abgeschlossen darf das Projekt erst gelten, wenn die Anwendung vollständig dokumentiert ist.



Spezifikation und Implementierung sind selten deckungsgleich. Problematisch ist der nicht spezifizierte Teil der Implementierung.

Entschärfung gibt es den SAP-Hinweis 1771262, der neugierigen Lesern ans Herz gelegt sei, weil er in diesem Kontext skurril wirkt.

Man muss den Hebel schon früher ansetzen, denn in vielen Unternehmen existiert kein definierter Entwicklungsprozess. Es gibt kein Ticketsystem, keinen geordneten Transportprozess, kein Testkonzept und keine Erkenntnisse über die Auslastung der Entwickler. Wenn der Team- oder Projektleiter den Stand eines Projekts erfahren möchte, muss er den Entwickler fragen, was bislang geschehen ist – vor allem in der Urlaubszeit oder im Krankheitsfall nicht einfach, also genau dann, wenn der Projektleiter diese Information dringend braucht.

Quer durch die Evolution

Viele SAP-Entwickler können auf ein Berufsleben zurückblicken, das weit vor der Einführung der Standardsoftware begann. Meist haben sie schon für das Altsystem (oft R/2) programmiert. In R/3 mussten sie dann diverse Entwicklungsstufen erklimmen: vom Reporting (ohne Datenveränderung mit logischen Datenbanken und Extrakten) über Dialogprogrammierung, Objektorientierung bis hin zu Business Server Pages, Web Dynpro und – brandneu – SAPUI5. Die Benutzerschnittstellentechnik verlangt von Entwicklern, die sich bisher bestenfalls mit Berechtigungsprüfungen beschäftigt haben, dass sie nicht nur neue Funktionen, sondern auch aktuelle Sicherheitsstrategien erlernen, für die ihnen aber noch das Bewusstsein fehlt.

In den klassischen Techniken lauern ebenfalls Stolperfallen: Anwendungen prüfen Eingaben nicht auf Plausibilität und fangen unerwartete Einträge in das OK-Code-Feld nicht ab. Insgesamt geht man vom gutwilligen Anwender aus, dem man bösartige Systemmanipulationen nicht unterstellt. In derartig nachlässig zusammengeschusterten Programmen kann der Entwickler durch Herumspielen Effekte erzielen, von denen er hinterher selbst nicht weiß, dass er sie implementiert hat. Mit der Nase auf solche Lücken gestoßen, reagieren die Betroffenen oftmals mit Ablehnung („das kann ich ja nicht auch noch prüfen“). Dann muss man sie nicht nur davon überzeugen, dass sie das nicht nur können, sondern dass diese Aufgabe zu ihren wichtigsten zählt. Eine Anwendung, die solche einfachen Anforderungen nicht erfüllt, darf nicht den Status „fertig“ erhalten.

Es ist gar nicht notwendig, das Rad ständig neu zu erfinden – statt Dynpros „von Hand“ zu programmieren, können Entwickler das objektorientierte BUS Screen Framework nutzen. Je sparsamer sie selbst kodieren, desto weniger Fehler entstehen. Ein vernünftiges, applikationsübergreifendes Klassenmodell reduziert die Anzahl der Codezeilen deutlich. Und mit Unit-Tests versehen, können Tester solche Klassen jederzeit auf das Erfüllen beliebiger Anforderungen prüfen.

Leider wird das Einführen objektorientierter Programmierung zusätzlich dadurch erschwert, dass Key-User oder Modulverantwortliche gern selbst per Debugger nach Fehlern suchen. In prozeduralen Programmen geht das durchaus, in objektorientierten muss

Professionelle Testverfahren

Gute Anwendungsentwicklung benötigt geeignete Testverfahren, die – wie jeder Geschäftsprozess – verbindlich festgelegt und für jede Produktivsetzung obligatorisch sein müssen.

Unit-Tests arbeiten alle gewünschten Funktionen in Positivtests, alle bekannten sowie denkbaren unerwünschten Funktionen in Negativtests ab. Das führt dazu, dass die gesamte Spezifikation einer Anwendung in Form von Testmethoden verfügbar ist. Sie lässt sich mit allen ihren Unterroutinen auf Erfüllung der Vorgaben testen. So kann es nicht passieren, dass ein Entwickler mit einer kleinen Änderung bestehende Funktionen „kaputt programmiert“.

Weiterhin gehört es zu einem guten, modernen Stil, dass jede Fehlermeldung ihren Niederschlag in Testmethoden findet. Der Fehler wird reproduzierbar nachgestellt und automatisch bei jedem Test durchlaufen. Das Wiederauftreten eines Fehlers durch eine Codeänderung würde jederzeit auffallen.

Ein korrektes Funktionieren aller Subroutinen stellt noch nicht sicher, dass ihre Zusammenarbeit reibungslos funktioniert. Daher müssen die Entwickler das Programm auch als Ganzes testen. SAP stellt Werkzeuge dafür zur Verfügung. eCATT etwa erlaubt das Anlegen und wiederholte Ausführen kompletter Testszenarien für ein einzelnes Programm, sogar über Systemgrenzen hinweg. So kann ein Testskript eine Anwendung auf dem Entwicklungssystem mit den Daten des Testsystems untersuchen.

Die letzten Untersuchungen beschäftigen sich mit der Systemintegration und dem Überprüfen der Benutzerakzeptanz (User Acceptance Tests). Da sie nicht automatisierbar sind, müssen sie die Key-User persönlich erledigen. Die Tests ermitteln, ob der Anwender mit dem fertiggestellten und korrekt funktionierenden Programm arbeiten will und kann und ob sich die Applikation harmonisch in alle benachbarten Prozesse einfügt. Die Anwender müssen diese Tests definieren, dokumentieren und ständig verbessern.

Ein gutes Prototyping und das frühzeitige Einbinden der Key-User in den Entwicklungsprozess (Stichwort: Handling-Mockup, ein Prototyp ohne echte Funktionen) verhindert das Auftreten unangenehmer Überraschungen.

man die Beziehungen zwischen den Objekten kennen, um die Programme zu verstehen. Hier zeigt sich, wie wichtig eine umfassende Dokumentation ist, die derartiges Wissen vermittelt. Auch die bei einigen Kunden praktizierte Unsitte, überflüssige Codezeilen nicht zu löschen, sondern lediglich unter Nennung der Ticketnummer auszukommentieren, hat zur Folge, dass sich der Code praktisch nicht mehr warten lässt, was Fehler geradezu provoziert. SAP bietet eine ziemlich gute Versionsverwaltung; Programme werden nicht dadurch besser, dass man die Versionshistorie im Code selbst abbildet.

Browser: Einfallstor für Schadcode

Bei den browsergestützten Techniken hat man es nicht mehr nur mit den bekannten Kollegen im Haus zu tun, sondern mit anonymen Anwendern. Die Adresszeile des Browsers ist ein Einfallstor für Manipulationen und eröffnet zahlreiche Wege, das Programmverhalten zu beeinflussen. Wie das OK-Code-Feld im SAP GUI wird die Adressleiste des Browsers viel zu selten als Feld erkannt, dessen Eingaben die Applikation auf Plausibilität prüfen muss. Potenzielle Sicherheitslücken existieren zahlreich, hier einige Beispiele:

– der Klassiker: *CALL TRANSACTION* führt keinerlei Berechtigungsprüfungen durch;
– noch schlimmer: fehlende Berechtigungsprüfungen in RFC-fähigen Funktionsbausteinen;
– hart kodierte Berechtigungen wie *if sy-uname = …*, mit denen Entwickler Berechtigungsprüfungen umgehen, um „ungestört“ in produktiven Umgebungen zu testen;
– syntaktische Manipulation von SQL-Abfragen (SQL Injection);
– fehlende Trennung zwischen Steuerzeichen, Befehlen und Daten;
– direktes Verwenden erratbarer Ressourcen statt pseudonymisierender IDs;
– fehlerhaft implementierte *ASSERT*-Kommandos;
– fehlerhaft implementiertes generisches Coding;
– Manipulation von HTML-Seiten durch Einschleusen von Code;
– Manipulation der Adresszeile im Browser;
– Cross Site Scripting.

Mit vielen dieser Begriffe können altgediente SAP-Entwickler wenig anfangen, was es erschwert, in ihnen ein Bewusstsein für diese Lücken zu wecken. Sie haben oft schon genug damit zu tun, sich in objektorientierte Konzepte, Model-View-Controller-Paradigmen (MVC) und Ähnliches einzuarbeiten. Zudem sehen sich Programmierer gern als kreative Köpfe und wehren sich mit aller Macht gegen die Einsicht, dass Softwareentwicklung ein industrieller Prozess ist, der entsprechenden Regeln unterliegt. Besonders gegen das Monitoring der Arbeitsfortschritte und -ergebnisse sträuben sich viele Entwickler.

Die Sicherheitsvorfälle in SAP-Systemen und deren Umfeld nehmen in Zahl und Folgenschwere deutlich zu, vor allem durch das Öffnen der Anwendungen für das Internet. Das Bewusstsein vieler Verantwortlicher hinkt dieser Entwicklung weit hinterher. Ein Unternehmen, das seine Daten ungeschützt allen potenziellen Angreifern vor die Nase hält, darf sich nicht wundern, wenn Böswillige ihre Chance wittern.

Wer die Sicherheit seiner Applikationen erhöhen will, sollte sich nicht davon abschrecken lassen, dass er in vielen Fällen die IT-Prozesse komplett umbauen muss. Wenn die IT zuverlässig arbeiten und strengen Audits standhalten soll, müssen sich alle Beteiligten bewegen. Das heißt: Kritik annehmen, Verantwortlichkeiten neu verteilen, Geschäftsprozesse nachvollziehbar definieren und permanent beobachten. (jd)

Ralf Wenzel

ist Inhaber der Unternehmensberatung Heuristika in Hamburg.

Daniel Lehmann

ist Geschäftsführer der General Data GmbH in Bielefeld.

Alle Links: www.ix.de/ix1410108

Die SAP-Entwicklungsumgebung

Eine Besonderheit zeichnet das SAP-Systems aus: Die Entwicklungsumgebung ist „mit sich selbst“ programmiert, was konkret bedeutet, dass das System grundsätzlich nicht zwischen Anwendungs- und Systemprogrammierung unterscheidet, der Kernel führt jeden Code aus. Dadurch eröffnen sich ungeahnte Möglichkeiten für die Softwareentwicklung. Programme können sich selbst verändern oder neue Programme zur Laufzeit erzeugen, ausführen und wieder entfernen (Code Injection), ohne Spuren zu hinterlassen.

Hinter der dynamischen Programmierung steht eine lange Tradition. Der Begriff „Dynpro“ – der für eine Bildschirmmaske verwendet wird – stammt aus der Anfangszeit dieser Idee und ist die Abkürzung für „dynamisches Programm“. Sie beschreibt eine Technik, die viel mehr umfasst als einen Screen. SAP war zu Recht stolz auf diese – in den 70er-Jahren völlig neue – Methode, die bereits die prozedurale Programmierung ablöste.

Die darin liegenden Möglichkeiten lassen sich jedoch missbrauchen. Wer in der Lage ist, Code in ein System einzuschleusen, kann hier praktisch alles tun, insbesondere in einem SAP-System, denn das erledigt auch die Berechtigungs- und Benutzerverwaltung mit den Mitteln der Entwicklungsumgebung (und in der Sprache ABAP). Die enge Anbindung an die Datenbank – die sich weitgehend aus SAP heraus verwalten lässt – erlaubt den uneingeschränkten Zugriff auf alle Daten, einschließlich Benutzerdaten, Berechtigungen, Programme und sämtliche Protokolle. Die Anbindung an das Betriebssystem erschließt darüber hinaus den Zugang zum Dateisystem.

Jeder Entwickler hat grundsätzlich die Mittel zur Verfügung, um alle verbundenen Systeme uneingeschränkt zu manipulieren. Er benötigt dazu zwar umfangreiche Kenntnisse, jedoch gibt es keine Möglichkeit, Missbrauch sicher auszuschließen. In der Zeit des Closed-Shop-Betriebes der Rechenzentren erschien das möglicherweise als akzeptables Risiko. Man wusste um die Umstände und gewährte zum Beispiel ausscheidenden Mitarbeitern keinen Zugriff mehr auf das System.

Volle Kontrolle über alles

Heute sind SAP-Anwendungen von außen zugänglich, und damit stehen die Sicherheitslücken auch fremden Entwicklern offen. In einem Installationspaket eines SAP-zertifizierten Softwareanbieters fand sich beispielsweise eine Funktion namens *Z_ABAP_INSTALL_AND_RUN*. Gedacht war sie zur variablen Datenabfrage aus einem Subsystem. Dass sie Scheunentore aufstieß, weil sich auf diesem Wege beliebiger Code einschleusen und ausführen lässt, wussten weder die Entwickler noch die Administratoren des Kunden. In Absprache mit der Geschäftsleitung ließ sich demonstrieren, dass man innerhalb von zehn Minuten unter ausschließlicher Verwendung dieser Funktion die volle Kontrolle über das System übernehmen kann.

Andere Drittanbieter benutzen eine undokumentierte Funktion zum Unkenntlichmachen ihres Sourcecodes. Eine so geschützte Software lässt sich naturgemäß nicht auf Sicherheitslücken prüfen. Das bleibt einigen Spezialisten vorbehalten, die wissen, wie man den Code wieder lesbar macht. Solche Programme enthalten zuweilen „Wartungszugänge“ (Backdoors), die dem Anbieter einen Zugriff auf das System an allen Berechtigungsprüfungen vorbei ermöglichen.