Lebenszyklus von Systemen

Das 14. Kapitel hat sich das Ziel gesetzt, dass Informationssicherheit ein fester Bestandteil über den gesamten Lebenszyklus von Informationssystemen ist. Es knüpft an die beiden Kapitel 12 (Betriebssicherheit, Maschinenbau 6/2019) und Kapitel 13 (Kommunikationssicherheit, Maschinenbau 7/2019) an und ergänzt diese optimal.

Bereits bei der Beschaffung von neuen Systemen gilt es die Informationssicherheit zu berücksichtigen. Dazu gilt es nicht nur theoretisch zu planen und Anforderungen zu definieren, sondern auch Tests und Simulationen zu planen. Gerade bei kritischen Systemen lohnt sich dieser Zeitaufwand. Kann das System in der eigenen Umgebung betrieben werden? Wie sehen die Schnittstellen aus? Wie kann das System überwacht werden? Wie wird es gesichert? Dies sind einige Fragen, die beantwortet werden sollten. Weiter gilt es die Fragen nach dem erwarteten Sicherheitsniveau, den Anforderungen an die Vertraulichkeit, Verfügbarkeit und Integrität der betroffenen Werte sowie notwendige Prozesse (oder Anpassungen) anzuschauen. Alle erwähnten Punkte sollten schriftlich festgehalten werden. Dazu gehört auch eine Risiko-Analyse (Maschinenbau 8/2018). Auf Basis dieser Resultate kann dann das weitere Vorgehen definiert werden.

Sicherung der Informationen

Ergänzend zur Kommunikationssicherheit verlangt das Kapitel A.14.1.2 die Sicherung der Informationen, welche in öffentlichen Netzwerken übertragen werden. Dies beginnt bereits bei der Authentifizierung der Gegenstelle. Oft werden dazu Zertifikate verwendet (Maschinenbau 4/2019). Damit kann einwandfrei gewährleistet werden, dass mit der richtigen Gegenstelle kommuniziert wird. Aber nicht nur technisch gilt es Massnahmen zu treffen, auch vertraglich gilt es sicherzustellen, dass alles korrekt läuft. Darin werden unter anderem die Verantwortlichkeiten, Verfügbarkeiten, Vorgehen bei Missbrauchsversuchen, Haftung und Notfallmassnahmen geregelt.

Der nächste Punkt schlägt in die gleiche Kerbe. Die Massnahme beschreibt schon alle Anforderungen: «Information, die an Transaktionen bei Anwendungsdiensten beteiligt ist, sollte so geschützt werden, dass unvollständige Übertragung, Fehlleitung, unbefugte Offenlegungen, unbefugte Vervielfältigungen oder unbefugte Wiederholung von Nachrichten verhindert ist.» Einige Möglichkeiten wurden bereits weiter oben beschrieben. Die Kryptographie liefert Möglichkeiten, die Transaktion zu verschlüsseln, die involvierten Stellen eindeutig zu identifizieren sowie eine Auswertung zu ermöglichen. Zudem empfiehlt es sich, diese Logdaten an einem geschützten Ort, das heisst nicht auf dem verwendeten Gerät, sondern in einem weiteren System zu sichern.

Entwicklung

Das zweite Kapitel (A.14.2) verlangt Sicherheit in Entwicklungs- und Unterstützungsprozessen. In einem ersten Schritt gilt es Richtlinien zur sicheren Entwicklung zu definieren. Dazu gehören die Umgebung, in welcher Software-Entwickler arbeiten, welches Wissen sie mitbringen, welche Methodik sie anwenden (Begriffe dazu: Wasserfall, Prototyping, Agil), welche Anforderungen an die Programmiersprache gelten (zum Beispiel welche Befehle nicht verwendet werden dürfen), wie Entwürfe erstellt, welche Milestones es gibt (zum Beispiel Abnahmekontrollen), wie der Code aufbewahrt wird (Repository), wie Versionen gesichert werden und wie Schwachstellen verhindert werden. Wer nun meint, diese Anforderungen gelten nur, wenn Software entwickelt wird, wird mit einem kurzen Schlusssatz enttäuscht: Die Entwicklung kann auch innerhalb von Anwendungen stattfinden (Büroanwendungen, Skripting, Browser und Datenbanken). Dies wird praktisch auf jedes Unternehmen zutreffen und macht es schwierig, diese Anforderungen auszuschliessen. Aus meiner Erfahrung akzeptieren zirka die Hälfte der Zertifizierungsstellen trotzdem den Ausschluss dieses Controls.

Im Internet sind zahlreiche Quellen mit Tipps und Tricks vorhanden. Bei der Entwicklung von Web-Applikationen hat sich OWASP etabliert. Das Open Web Application Security Project (www.owasp.org) ist vor allem mit seinen Top-10-Schwachstellen bekannt geworden. Zu jeder aufgelisteten Schwachstelle gibt es für verschiedene Programmiersprachen Anleitungen, wie diese verhindert werden können.

Weiter gehören auch Vorgaben an eine sichere Entwicklungsumgebung dazu. Neben technischen Vorgaben ist auch die Vertrauenswürdigkeit der involvierten Personen zu prüfen. Entwicklungsumgebungen sind strikt von produktiven Systemen zu trennen. Der Zugang zur Entwicklung ist auf wenige Personen einzuschränken. Wird die Entwicklung ausgelagert, was in vielen Unternehmen der Fall sein dürfte, gilt es den Prozess zu beaufsichtigen und zu überwachen. Jede Änderung gilt es schriftlich festzuhalten, nach der Umsetzung ausgiebig zu testen und durch die entsprechenden internen Stellen abzunehmen. Für den schlimmsten Fall empfehlen sich Escrow-Verträge. Wird das Unternehmen insolvent, wird der Source-Code herausgegeben und damit kann ein anderes Unternehmen zur Weiterarbeit beauftragt werden.

Bild: Archiv

Unbefugte Vervielfältigungen oder unbefugte Wiederholung von Nachrichten verhindern.

Berechtigungen definieren

Alle Systemänderungen gilt es immer zu sichern. Repositories machen dies in der Regel automatisch. Aber auch für «kleine» Skripts gilt diese Anforderung. Es muss sichergestellt sein, dass Änderungen nachvollziehbar sind. Trotz der oft gehörten Meinung «Der Code dokumentiert sich selbst», gehört eine Dokumentation dazu. Vermutlich ist es allen schon so ergangen, dass ein super grossartiges Skript geschrieben wurde, das eine Zeit später angepasst werden musste und man selber nicht mehr wusste, was es

denn nun genau macht. Daher, auch wenn es mühsam ist, die Dokumentation gehört zwingend dazu. Dieser Norm-Punkt listet übrigens insgesamt zwölf Anforderungen auf. Dazu gehören unter anderem das Definieren von Berechtigungen, dass nur befugte Benutzer Änderungen melden dürfen, Kontrollen vor der Änderung, Schwachstellenprüfung/Code-Reviews vor der Veröffentlichung, angepassten Genehmigungsprozesse, Prüfpfade nach der Umsetzung und der anschliessenden Anpassung der Dokumentation.

Aber nicht nur bei Code-Änderungen gilt es Prozesse zu definieren. Auch die Aktualisierung von Systemen gehört dazu. Die Norm hat dazu ein eigenes Unterkapitel definiert (A.14.2.3). Nicht nur Betriebssysteme ändern sich, sie dies Windows, Macs oder Linux-Derivate. Bei Windows 10 ändert sich das Betriebssystem alle halben Jahre (ausser es wird zurückgestellt). Bei Servern ist dies zum Glück nicht so oft. So gilt es sicherzustellen, dass alle Applikationen nach der Aktualisierung noch laufen. Bei Servern stehen oft Redundanzen zur Verfügung. So kann ausgiebig getestet werden. Bei Clients lohnt es sich, eine Testgruppe zu bestimmen und erste Erfahrungen zu sammeln.

Die erwähnten Vorgaben, wie auch die Beschränkung von Änderungen an Softwarepaketen, gilt es in Grundsätzen fest zu halten. Wie werden neue Systeme evaluiert, getestet und in den Betrieb übernommen? Diese Dokumentation muss immer aktuell sein.

Von A bis Z sicher betreiben

Werden Tests gemacht, egal bei neuen Systemen oder in der Entwicklung, gilt es Testdaten zu schützen. Idealerweise werden dazu pseudonymisierte oder noch besser anonymisierte Daten verwendet. Bei der Pseudonymisierung wird das Ersetzen des Namens und anderer Identifikationsmerkmale durch ein Kennzeichen, die Bestimmung des Betroffenen ausgeschlossen oder wesentlich erschwert. Das heisst, mit entsprechendem Wissen kann wieder auf die ursprünglichen Daten zurück gelangt werden. Daher gilt es diese Kombination sicher aufzubewahren. Bei der Anonymisierung ist dies nicht mehr möglich, da in der Regel eine Einweg-Funktion verwendet wird. Aus dem Ergebnis kann nicht mehr auf die ursprünglichen Daten zurück gelangt werden.

Das Kapitel 14 ist sehr umfassend, beschreibt den Zyklus von neuen Systemen von der Entwicklung, dem Testen bis zur Veröffentlichung. Werden alle Punkte sauber dokumentiert und umgesetzt, können Neuentwicklungen, wie auch die Pflege bestehender Systeme optimal in die normalen Betriebsprozesse integriert und damit die System-Sicherheit massiv erhöht werden.