Bundesamt
für Sicherheit in der
Informationstechnik

CON: Konzepte und Vorgehensweisen

CON.8: Software-Entwicklung

1 Beschreibung

1.1 Einleitung

Viele Institutionen stehen vor Herausforderungen, die sie nicht mehr hinreichend mit einem fertigen, unangepassten Softwareprodukt lösen können. Sie benötigen hierzu Software-Lösungen, die auf ihre individuellen Anforderungen hin angepasst sind. Beispiele hierfür sind hochspezifische Software-Lösungen für Branchenspezialisten (wie zur Steuerung von Produktionsanlagen) oder an die eigenen Geschäftsprozesse angepasste IT-Anwendungen (wie Content-Management-Systeme oder Identity-Management-Systeme). Aber auch Altsysteme, die nicht mehr vom ursprünglichen Hersteller weitergepflegt werden, können individuell weiterentwickelt werden.

Hierbei kann (Individual-) Software durch die Institution selbst oder von einem Dritten entwickelt werden. Die Software-Entwicklung nimmt dabei eine zentrale Rolle ein, wenn aus den Anforderungen der Institution ein Programm-Code entwickelt bzw. angepasst wird. Hierbei ist es von essentieller Bedeutung, dass die Informationssicherheit über den gesamten Software-Entwicklungsprozess hinweg berücksichtigt wird und nicht erst in einer späteren Phase. Nur auf diese Weise kann die Informationssicherheit der zu entwickelnden Software-Lösung nachhaltig gewährleistet werden.

Software kann dabei im Rahmen von klassischen, in sich abgeschlossenen Projekten entwickelt werden, oder aber als kontinuierliche Tätigkeit ohne festes Ende. In beiden Fällen werden in der Praxis sehr häufig Werkzeuge aus dem Projektmanagement benutzt, um die Software-Entwicklung zu koordinieren und zu steuern. Deswegen wird in diesem Baustein der Begriff Projekt vermehrt verwendet und auch nicht durchgehend zwischen einer projektbasierten und kontinuierlichen Entwicklung unterschieden, da sich die damit verbundenen Vorgehensweisen und Werkzeuge ähneln.

1.2 Zielsetzung

Der Baustein beschäftigt sich mit allen relevanten Sicherheitsaspekten, die von Institutionen bei der Eigenentwicklung von Software zu beachten sind. Hierzu wird betrachtet, wie eine Institution die Software-Entwicklung vorbereiten und durchführen kann. Es werden entsprechende Gefährdungen identifiziert und Anforderungen formuliert.

1.3 Abgrenzung und Modellierung

Der Baustein CON.8 *Software-Entwicklung* ist für den Informationsverbund einmal anzuwenden, wenn Software entwickelt werden soll.

Wird Software entwickelt, liegt sehr häufig ein Auftraggeber- und Auftragnehmer-Verhältnis vor. Im IT-Grundschutz spiegelt sich dieser Sachverhalt wider, indem der Baustein APP.7 *Entwicklung von*

Individualsoftware die Auftraggeberseite und der Baustein CON.8 *Software-Entwicklung* die Auftragnehmerseite umfassen. Die Anforderungen dieses Bausteins sind somit vom Auftragnehmer zu erfüllen. Die für die Software-Entwicklung relevanten Anforderungen (funktionale und nicht-funktionale Anforderungen, Anforderungen an die sichere Vorgehensweise sowie das Sicherheitsprofil) werden vom Auftraggeber im Rahmen des Bausteins APP.7 *Entwicklung von Individualsoftware* erhoben.

Der Baustein stellt keine vollständige Anleitung oder generelle Vorgehensweise für die Entwicklung von Software dar, sondern er konzentriert sich auf die relevanten Aspekte der Informationssicherheit bei der Software-Entwicklung. Der Baustein umfasst ferner keine Aspekte zum Patch- und Änderungsmanagement. Hierzu ist der Baustein OPS.1.1.3 *Patch- und Änderungsmanagement* anzuwenden.

Die Abnahme und hiermit verbundenen Tests von eigenentwickelter bzw. beauftragter Software werden in dem Baustein OPS.1.1.6 *Software-Tests und Freigaben* geregelt. Darüber hinaus gehende Aspekte zu Tests im Rahmen der Software-Entwicklung werden in diesem Baustein CON.8 *Software-Entwicklung* behandelt.

Der Baustein ORP.5 *Compliance Management* muss mit betrachtet werden, da über diesen Baustein geregelt wird, wie die gesetzlichen und institutsinterne Vorgaben, sowie Anforderungen des Kunden berücksichtigt werden.

Umfasst die Software-Entwicklung kryptographische Aspekte, dann sind die relevanten Anforderungen aus dem Baustein CON.1 *Kryptokonzept* zu berücksichtigen.

2 Gefährdungslage

Folgende spezifische Bedrohungen und Schwachstellen sind für den Baustein CON.8 *Software-Entwicklung* von besonderer Bedeutung.

2.1 Auswahl eines ungeeigneten Vorgehensmodells

Vorgehensmodelle strukturieren und planen den Ablauf der Software-Entwicklung, indem bestimmte Handlungsschritte und deren Abfolge vorgegeben werden. Wird ein ungeeignetes Vorgehensmodell bei der Software-Entwicklung ausgewählt, kann der Verlauf der Entwicklung und das damit verbundene Entwicklungsprojekt erheblich gestört werden. Je nachdem, wie das gewählte Modell ausgeprägt und wie umfangreich das Entwicklungsvorhaben ist, könnten entweder wichtige Aspekte vernachlässigt oder irrelevante Aspekte übermäßig fokussiert werden. Beide genannten Probleme erhöhen den Arbeitsaufwand im Projektmanagement und schränken die produktive Arbeit ein.

Wird gar kein Vorgehensmodell verwendet, erhöht sich die Gefahr, dass relevante Aspekte, die insbesondere die Informationssicherheit betreffen, in der Software-Entwicklung nicht in geeigneter Art und Weise berücksichtigt werden. Dies kann dazu führen, dass relevante Sicherheitsfunktionen überhaupt nicht implementiert oder getestet werden, sodass die entwickelte Software nicht den Sicherheitsanforderungen entspricht.

2.2 Auswahl einer ungeeigneten Entwicklungsumgebung

Wird eine Entwicklungsumgebung ungeeignet oder von jedem Mitarbeiter individuell ausgewählt, können dringend benötigte Funktionen fehlen oder nicht in ausreichender Form implementiert sein. Weiterhin kann eine ungeeignete Entwicklungsumgebung auch Fehler oder Schwachstellen aufweisen, die erhebliche Störungen im Verlauf der Software-Entwicklung verursachen können.

Wenn keine bestimmte Entwicklungsumgebung ausgewählt und vorgegeben wird, arbeiten verschiedene Entwickler möglicherweise mit unterschiedlichen, selbst gewählten Werkzeugen an der Software und können dadurch Kompatibilitätsprobleme verursachen. Beispielweise können unterschiedliche Compiler solche Kompatibilitätsprobleme auslösen.

2.3 Fehlende oder unzureichende Qualitätssicherung des Entwicklungsprozesses

Durch eine fehlende oder unzureichende Qualitätssicherung während der Software-Entwicklung kann sich das Entwicklungsvorhaben verzögern oder sogar gänzlich scheitern. Wenn nicht geprüft wird, ob die eigenentwickelte Software sicher implementiert wird, drohen Schwachstellen in der ausgelieferten Software.

Ist die Qualitätssicherung nicht fest im Entwicklungsprozess verankert, können Fehler und Manipulationen in der Konzeption oder Implementierung unter Umständen nicht erkannt werden. Dabei sollte die Aufmerksamkeit nicht nur selbst entwickelten Bestandteilen, sondern gerade auch externen Beiträgen und übernommenen Bestandteilen gelten.

2.4 Fehlende oder unzureichende Dokumentation

Wird die Software in der Konzeptions- oder Entwicklungsphase nicht oder nur unzureichend dokumentiert, kann dies dazu führen, dass eventuelle Fehler nur verzögert oder gar nicht diagnostiziert und behoben werden können. Wird die Entwicklung ungeeignet dokumentiert, kann die Software außerdem später nur mit hohem Aufwand aktualisiert, angepasst oder erweitert werden.

Bei unzureichender Administratoren- oder Benutzerdokumentation könnte die Software im produktiven Betrieb fehlerhaft verwaltet oder bedient werden. Dies kann beispielsweise den IT-Betrieb stören, falsche Arbeitsergebnisse verursachen oder den Arbeitsablauf verzögern.

2.5 Unzureichend gesicherter Einsatz von Entwicklungsumgebungen

Wenn die Entwicklungsumgebung unzureichend gesichert wird, kann die zu produzierende Software möglicherweise manipuliert werden. Solche Manipulationen können dadurch nachträglich nur schwer erkannt werden.

Wenn nicht bekannt ist, welche Benutzer zu welchem Zeitpunkt auf die Entwicklungsumgebung zugreifen können und konnten, kann die Software anonym manipuliert werden. Sofern die manipulierten Teile der Software entdeckt werden, kann dann unter Umständen nicht nachvollzogen werden, welcher Benutzer sie manipuliert hat.

Bei einer fehlenden oder unzureichenden Versionsverwaltung des Quellcodes ist es nicht möglich, Änderungen zuverlässig nachzuvollziehen sowie vorherige und bereits funktionierende Versionen der Software wiederherzustellen.

Wenn Quellcodes unzureichend vor versehentlichen oder absichtlichen Änderungen geschützt werden, können Teile eines Quellcodes oder sogar der gesamte Quellcode beschädigt werden und die bereits eingebrachte Arbeitsleistung verloren gehen.

Wird der Quellcode bzw. die Versionsverwaltung nicht hinreichend vor einem Datenverlust geschützt, folgen daraus verschiedene Gefährdungen, unabhängig davon, ob der Datenverlust z. B. durch einen technischen Defekt oder durch menschliches Versagen ausgelöst wird. Möglicherweise kann die Software überhaupt nicht mehr weiter entwickelt werden, da der Datenbestand gänzlich fehlt, oder es ist nur ein veralteter und möglicherweise fehlerhafter Zwischenstand verfügbar, der nur mit sehr hohen Aufwänden verwendet werden kann.

2.6 Software-Konzeptionsfehler

Je umfangreicher eine Software vom Funktionsumfang wird, umso umfangreicher wird häufig auch ihr Programm-Code. Wenn der Programm-Code nicht durch geeignete Maßnahmen strukturiert wird und wenn er nicht auf einer angemessen Software-Architektur basiert, dann kann er zumeist nur sehr schwer gewartet werden. So können Sicherheitslücken nur schwer geschlossen oder veraltete Programm-Teile nur mit sehr hohen Aufwand ausgetauscht werden, wenn sich z.B. der Schutzbedarf der verarbeiteten Daten ändert und somit auch die Sicherheitsanforderungen an die Software.

Software-Konzeptionsfehler erschweren hierbei nicht nur die Wartung der Software, sondern sie

können selbst zu Sicherheitslücken und Gefährdungen führen. Ist der Programm-Code nicht sinnvoll strukturiert und die Software-Architektur nicht nachvollziehbar dokumentiert, dann können konzeptionelle Fehler in Software-Tests nur sehr schwer identifiziert werden. In Folge dessen können auf den unterschiedlichsten Ebenen der Software Sicherheitslücken bestehen.

Software-Konzeptionsfehler sind in der Praxis häufig historisch bedingt, indem z.B. Altsysteme für Aufgaben und Umgebungen eingesetzt werden, für diese sie zuerst gar nicht konzeptioniert worden sind. Auch wurden bei der Software-Entwicklung von sehr alten Anwendungen Aspekte wie Wartbarkeit und Modifizierbarkeit nicht in dem erforderlichen Maße berücksichtigt, wie es heute dem Stand der Technik entspricht.

2.7 Fehlendes oder unzureichendes Test- und Freigabeverfahren

Wird neue Software nicht ausreichend getestet und freigegeben, können Fehler in der Software nicht erkannt werden. Solche Fehler gefährden nicht nur den produktiven Einsatz und die Informationssicherheit der neuen Software selbst, sondern unter Umständen auch andere Anwendungen und IT-Systeme in der Produktivumgebung.

Werden Sicherheitsfunktionen bzw. die grundlegenden Sicherheitsanforderungen nicht getestet, ist nicht sichergestellt, dass die entwickelte Software den Sicherheitsanforderungen der einsetzenden Institution genügt. In Folge dessen könnten schützenswerte Informationen offen gelegt, manipuliert oder zerstört werden, indem z.B. unbefugte Dritte aufgrund mangelhafter Authentifizierungsfunktionen auf die Software zugreifen.

2.8 Software-Tests mit Produktivdaten

Wenn neue Software mit Produktivdaten getestet wird, könnten eventuell nicht befugte Personen hierbei vertrauliche Informationen einsehen, wie besonders schützenswerte personenbezogene Daten.

Wird nicht mit Kopien der Produktivdaten getestet, sondern mit den Originalen (z.B. bei Datenbanksystemen) entstehen noch umfangreichere Gefährdungen:

- Fehlfunktionen von Software während des Testens können dazu führen, dass neben der Vertraulichkeit der Produktivdaten auch deren Integrität oder Verfügbarkeit beeinträchtigt wird.
- Ungewollte Veränderungen an Produktivdaten können auch dadurch entstehen, dass die Software fehlerhaft getestet oder bedient wird. Möglicherweise wird diese Veränderung nicht zeitnah festgestellt. Solche Fehler können sich auch auf andere IT-Anwendungen auswirken, die auf die gleichen Datenbestände zugreifen.

Diese Umstände werden sehr häufig dadurch verschärft, dass während die Software getestet wird, nicht der Schutz der Testdaten im Vordergrund steht, sondern, ob die Software sich wie gewünscht, bzw. durch die Anforderungen definiert, verhält.

3 Anforderungen

Im Folgenden sind die spezifischen Anforderungen des Bausteins CON.8 *Software-Entwicklung* aufgeführt. Grundsätzlich ist der Fachverantwortliche für die Erfüllung der Anforderungen zuständig. Der Informationssicherheitsbeauftragte (ISB) ist bei strategischen Entscheidungen stets einzubeziehen. Außerdem ist der ISB dafür zuständig, dass alle Anforderungen gemäß dem festgelegten Sicherheitskonzept erfüllt und überprüft werden. Zusätzlich kann es noch andere Rollen geben, die weitere Zuständigkeiten bei der Erfüllung von Anforderungen haben. Diese sind dann jeweils explizit in eckigen Klammern in der Überschrift der jeweiligen Anforderungen aufgeführt.

Zuständigkeiten	**Rollen**
Grundsätzlich zuständig	Fachverantwortliche

Weitere Zuständigkeiten	Tester, Zentrale Verwaltung, IT-Betrieb, Entwickler

3.1 Basis-Anforderungen

Die folgenden Anforderungen MÜSSEN für den Baustein CON.8 *Software-Entwicklung* vorrangig erfüllt werden:

CON.8.A2 Auswahl eines Vorgehensmodells (B)

Ein geeignetes Vorgehensmodell zur Software-Entwicklung MUSS festgelegt werden. Anhand des gewählten Vorgehensmodells MUSS ein Ablaufplan für die Software-Entwicklung erstellt werden. Die Sicherheitsanforderungen des Auftraggebers an die Vorgehensweise MÜSSEN im Vorgehensmodell integriert werden.

Das ausgewählte Vorgehensmodell, einschließlich der festgelegten Sicherheitsanforderungen, MUSS eingehalten werden.

Das Personal SOLLTE in der Methodik des gewählten Vorgehensmodells geschult sein.

CON.8.A3 Auswahl einer Entwicklungsumgebung (B)

Eine Liste der erforderlichen und optionalen Auswahlkriterien für eine Entwicklungsumgebung MUSS vom Fachverantwortlichen für die Software-Entwicklung erstellt werden. Die Entwicklungsumgebung MUSS anhand der vorgegebenen Kriterien ausgewählt werden.

CON.8.A4 ENTFALLEN (B)

Diese Anforderung ist entfallen.

CON.8.A5 Sicheres Systemdesign (B)

Folgende Grundregeln des sicheren Systemdesigns MÜSSEN in der zu entwickelnden Software berücksichtigt werden:

- Grundsätzlich MÜSSEN alle Eingabedaten vor der Weiterverarbeitung geprüft und validiert werden.
- Bei Client-Server-Anwendungen MÜSSEN die Daten grundsätzlich auf dem Server validiert werden.
- Die Standardeinstellungen der Software MÜSSEN derart voreingestellt sein, dass ein sicherer Betrieb der Software ermöglicht wird.
- Bei Fehlern oder Ausfällen von Komponenten des Systems DÜRFEN NICHT schützenswerte Informationen preisgegeben werden.
- Die Software MUSS mit möglichst geringen Privilegien ausgeführt werden können.
- Schützenswerte Daten MÜSSEN entsprechend der Vorgaben des Kryptokonzepts verschlüsselt übertragen und gespeichert werden.
- Zur Benutzer-Authentisierung und Authentifizierung MÜSSEN vertrauenswürdige Mechanismen verwendet werden, die den Sicherheitsanforderungen an die Anwendung entsprechen.
- Falls zur Authentifizierung Passwörter gespeichert werden, MÜSSEN diese mit einem sicheren Hashverfahren gespeichert werden.
- Sicherheitsrelevante Ereignisse MÜSSEN in der Art protokolliert werden, dass sie im Nachgang ausgewertet werden können.
- Informationen, die für den Produktivbetrieb nicht relevant sind (z. B. Kommentare mit Zugangsdaten für die Entwicklungsumgebung), SOLLTEN in ausgeliefertem Programmcode und ausgelieferten Konfigurationsdateien entfernt werden.

Das Systemdesign MUSS dokumentiert werden. Es MUSS überprüft werden, ob alle Sicherheitsanforderungen an das Systemdesign erfüllt wurden.

CON.8.A6 Verwendung von externen Bibliotheken aus vertrauenswürdigen Quellen (B)

Wird im Rahmen des Entwicklungs- und Implementierungsprozesses auf externe Bibliotheken zurückgegriffen, MÜSSEN diese aus vertrauenswürdigen Quellen bezogen werden. Bevor externe Bibliotheken verwendet werden, MUSS deren Integrität sichergestellt werden.

CON.8.A7 Durchführung von entwicklungsbegleitenden Software-Tests [Tester, Entwickler] (B)

Schon bevor die Software im Freigabeprozess getestet und freigegeben wird, MÜSSEN entwicklungsbegleitende Software-Tests durchgeführt und der Quellcode auf Fehler gesichtet werden. Hierbei SOLLTEN bereits die Fachverantwortlichen des Auftraggebers oder der beauftragenden Fachabteilung beteiligt werden.

Die entwicklungsbegleitenden Tests MÜSSEN die funktionalen und nichtfunktionalen Anforderungen der Software umfassen. Die Software-Tests MÜSSEN dabei auch Negativtests abdecken. Zusätzlich MÜSSEN auch alle kritischen Grenzwerte der Eingabe sowie der Datentypen überprüft werden.

Testdaten SOLLTEN dafür sorgfältig ausgewählt und geschützt werden. Darüber hinaus SOLLTE eine automatische statische Code-Analyse durchgeführt werden.

Die Software MUSS in einer Test- und Entwicklungsumgebung getestet werden, die getrennt von der Produktionsumgebung ist. Außerdem MUSS getestet werden, ob die Systemvoraussetzungen für die vorgesehene Software ausreichend dimensioniert sind.

CON.8.A8 Bereitstellung von Patches, Updates und Änderungen [Entwickler] (B)

Es MUSS sichergestellt sein, dass sicherheitskritische Patches und Updates für die entwickelte Software zeitnah durch die Entwickler bereitgestellt werden. Werden für verwendete externe Bibliotheken sicherheitskritische Updates bereitgestellt, dann MÜSSEN die Entwickler ihre Software hierauf anpassen und ihrerseits entsprechende Patches und Updates zur Verfügung stellen.

Für die Installations-, Update- oder Patchdateien MÜSSEN vom Entwickler Checksummen oder digitale Signaturen bereitgestellt werden.

CON.8.A9 ENTFALLEN (B)

Diese Anforderung ist entfallen.

CON.8.A10 Versionsverwaltung des Quellcodes [Entwickler] (B)

Der Quellcode des Entwicklungsprojekts MUSS über eine geeignete Versionsverwaltung verwaltet werden. Die Institution MUSS den Zugriff auf die Versionsverwaltung regeln und festlegen, wann Änderungen am Quellcode durch die Entwickler als eigene Version in der Versionsverwaltung gespeichert werden sollen. Es MUSS sichergestellt sein, dass durch die Versionsverwaltung alle Änderungen am Quellcode nachvollzogen und rückgängig gemacht werden können.

Die Versionsverwaltung MUSS in dem Datensicherungskonzept berücksichtigt werden. Die Versionsverwaltung DARF NICHT ohne Datensicherung erfolgen.

CON.8.A20 Überprüfung von externen Komponenten (B)

Unbekannte externe Komponenten (bzw. Programm-Bibliotheken), deren Sicherheit nicht durch etablierte und anerkannte Peer-Reviews oder vergleichbares sichergestellt werden kann, MÜSSEN auf Schwachstellen überprüft werden. Alle externen Komponenten MÜSSEN auf potentielle Konflikte überprüft werden.

Die Integrität von externen Komponenten MUSS durch Prüfsummen oder kryptographische Zertifikate überprüft werden.

Darüber hinaus SOLLTEN keine veralteten Versionen von externen Komponenten in aktuellen Entwicklungsprojekten verwendet werden.

3.2 Standard-Anforderungen

Gemeinsam mit den Basis-Anforderungen entsprechen die folgenden Anforderungen dem Stand der Technik für den Baustein CON.8 *Software-Entwicklung*. Sie SOLLTEN grundsätzlich erfüllt werden.

CON.8.A1 Definition von Rollen und Zuständigkeiten [Zentrale Verwaltung] (S)

Für den Software-Entwicklungsprozess SOLLTE ein Gesamtzuständiger benannt werden. Außerdem SOLLTEN die Rollen und Zuständigkeiten für alle Aktivitäten im Rahmen der Software-Entwicklung festgelegt werden. Die Rollen SOLLTEN dabei fachlich die nachfolgenden Themen abdecken:

- Requirements-Engineering (Anforderungsmanagement) und Änderungsmanagement,
- Software-Entwurf und -Architektur,
- Informationssicherheit in der Software-Entwicklung,
- Software-Implementierung in dem für das Entwicklungsvorhaben relevanten Bereichen, sowie
- Software-Tests.

Für jedes Entwicklungsvorhaben SOLLTE ein Zuständiger für die Informationssicherheit benannt werden.

CON.8.A11 Erstellung einer Richtlinie für die Software-Entwicklung (S)

Es SOLLTE eine Richtlinie für die Software-Entwicklung erstellt und aktuell gehalten werden. Die Richtlinie SOLLTE neben Namenskonventionen auch Vorgaben zu Elementen beinhalten, die verwendet bzw. nicht verwendet werden dürfen. Die relevanten Anforderungen aus diesem Baustein SOLLTEN in die Richtlinie aufgenommen werden. Die Richtlinie SOLLTE für die Entwickler verbindlich sein.

CON.8.A12 Ausführliche Dokumentation (S)

Es SOLLTEN ausreichende Projekt-, Funktions- und Schnittstellendokumentationen erstellt und aktuell gehalten werden. Die Betriebsdokumentation SOLLTE konkrete Sicherheitshinweise für die Installation und Konfiguration für Administratoren, sowie für die Benutzung des Produktes beinhalten.

Die Software-Entwicklung SOLLTE so dokumentiert sein, dass ein Fachexperte mithilfe der Dokumentation den Programm-Code nachvollziehen und weiterentwickeln kann. Die Dokumentation SOLLTE dabei auch die Software-Architektur und Bedrohungsmodellierung umfassen.

Die Aspekte zur Dokumentation SOLLTEN im Vorgehensmodell zur Software-Entwicklung berücksichtigt werden.

CON.8.A13 ENTFALLEN (S)

Diese Anforderung ist entfallen.

CON.8.A14 Schulung des Entwicklungsteams zur Informationssicherheit (S)

Die Entwickler und die übrigen Mitglieder des Entwicklungsteams SOLLTEN zu generellen Informationssicherheitsaspekten und zu den jeweils speziell für sie relevanten Aspekten geschult sein:

- Anforderungsanalyse,
- Projektmanagement allgemein sowie speziell bei der Software-Entwicklung„
- Risikomanagement bzw. Bedrohungsmodellierung in der Software-Entwicklung,
- Qualitätsmanagement und Qualitätssicherung,
- Modelle und Methoden für die Software-Entwicklung,
- Software-Architektur,
- Software-Tests,
- Änderungsmanagement sowie

- Informationssicherheit, Sicherheitsvorgaben in der Institution und Sicherheitsaspekte in speziellen Bereichen.

CON.8.A15 ENTFALLEN (S)

Diese Anforderung ist entfallen.

CON.8.A16 Geeignete Steuerung der Software-Entwicklung (S)

Bei einer Software-Entwicklung SOLLTE ein geeignetes Steuerungs- bzw. Projektmanagementmodell auf Basis des ausgewählten Vorgehensmodells verwendet werden. Das Steuerungs- bzw. Projektmanagementmodell SOLLTE in die Richtlinie zur Software Entwicklung integriert werden. Dabei SOLLTEN insbesondere die benötigten Qualifikationen beim Personal und die Abdeckung aller relevanten Phasen während des Lebenszyklus der Software berücksichtigt werden. Für das Vorgehensmodell SOLLTE ein geeignetes Risikomanagement festgelegt werden. Außerdem SOLLTEN geeignete Qualitätsziele für das Entwicklungsprojekt definiert werden.

CON.8.A21 Bedrohungsmodellierung (S)

In der Entwurfsphase der Software-Entwicklung SOLLTE eine Bedrohungsmodellierung durchgeführt werden. Hierzu SOLLTEN auf Basis des Sicherheitsprofils, des Anforderungskatalogs und der geplanten Einsatzumgebung bzw. Einsatzszenarios potentielle Bedrohungen identifiziert werden. Die Bedrohungen SOLLTEN hinsichtlich ihrer Eintrittswahrscheinlichkeit und Auswirkung bewertet werden.

CON.8.A22 Sicherer Software-Entwurf (S)

Der Software-Entwurf SOLLTE den Anforderungskatalog, das Sicherheitsprofil und die Ergebnisse der Bedrohungsmodellierung berücksichtigen. Im Rahmen des sicheren Software-Entwurfs SOLLTE eine sichere Software-Architektur entwickelt werden, auf deren Grundlage der Quellcode der Anwendung zu entwickeln ist. Hierbei SOLLTEN möglichst zukünftige Standards und Angriffstechniken berücksichtigt werden, damit die zu entwickelnde Software auch zukünftig leicht gewartet werden kann.

3.3 Anforderungen bei erhöhtem Schutzbedarf

Im Folgenden sind für den Baustein CON.8 *Software-Entwicklung* exemplarische Vorschläge für Anforderungen aufgeführt, die über das dem Stand der Technik entsprechende Schutzniveau hinausgehen und BEI ERHÖHTEM SCHUTZBEDARF in Betracht gezogen werden SOLLTEN. Die konkrete Festlegung erfolgt im Rahmen einer Risikoanalyse.

CON.8.A17 Auswahl vertrauenswürdiger Entwicklungswerkzeuge (H)

Zur Entwicklung der Software SOLLTEN nur Werkzeuge mit nachgewiesenen Sicherheitseigenschaften verwendet werden. An die Hersteller von Hardware oder Software SOLLTEN hinreichende Anforderungen zur Sicherheit ihrer Werkzeuge gestellt werden.

CON.8.A18 Regelmäßige Sicherheitsaudits für die Entwicklungsumgebung (H)

Es SOLLTEN regelmäßige Sicherheitsaudits der Software-Entwicklungsumgebung und der Software-Testumgebung durchgeführt werden.

CON.8.A19 Regelmäßige Integritätsprüfung der Entwicklungsumgebung [IT-Betrieb] (H)

Die Integrität der Entwicklungsumgebung SOLLTE regelmäßig mit kryptographischen Mechanismen entsprechend dem Stand der Technik geprüft werden. Die Prüfsummendateien und das Prüfprogramm selbst SOLLTEN ausreichend vor Manipulationen geschützt sein. Wichtige Hinweise auf einen Integritätsverlust SOLLTEN nicht in einer Fülle irrelevanter Warnmeldungen (false positives) untergehen.

4 Weiterführende Informationen

4.1 Wissenswertes

Die International Organization for Standardization (ISO) stellt Anforderungen zur Software-Entwicklung unter anderem in diesen Normen:

- ISO/IEC 27001:2013 Apendix A.14.2 „Security in development and support processes“ - Anforderungen an die Sicherheit in Entwicklungs- und Unterstützungsprozessen,
- ISO/IEC 25000:2014 „ Systems and software Quality Requirements and Evaluation - Guide to SQuaRE“ - ein genereller Überlick über die SQuaRE-Normen-Reihe,
- ISO/IEC 25001:2014 „Planning and management“ - Anforderungen an die Planung und das Management sowie
- ISO/IEC 25010:2011 „System and software quality models“ - Anforderungen an ein Qualitätsmodell und Leitlinien.

Das Infomation Security Forum (ISF) macht in seinem Standard „The Standard of Good Practice for Information Security“ im Kapitel „SD System Development“ Vorgaben an die sichere Software-Entwicklung,.

Das National Institute of Standards and Technologie gibt in seiner Special Publication 800-160 „Systems Security Engineering, Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems“ Anforderungen an ein sicheres Systemdesign.

Die Fachgruppe „Vorgehensmodelle für die betriebliche Anwendungsentwicklung“ der Gesellschaft für Informatik gibt in ihren Publikationen einen Überblick über aktuelle Informationen zur Anwendungsentwicklung.

Weiterführende Informationen zur Bedrohungsmodellierung können dem wissenschaftlichen Artikel „Bedrohungsmodellierung (Threat Modeling) in der Softwareentwicklung“ entnommen werden. Der Artikel wurde zu der Konferenz „Sicherheit 2010: Sicherheit, Schutz und Zuverlässigkeit)“ der Gesellschaft für Information veröffentlicht.

5 Anlage: Kreuzreferenztabelle zu elementaren Gefährdungen

Die Kreuzreferenztabelle enthält die Zuordnung von elementaren Gefährdungen zu den Anforderungen. Anhand dieser Tabelle lässt sich ermitteln, welche elementaren Gefährdungen durch welche Anforderungen abgedeckt sind. Durch die Umsetzung der aus den Anforderungen abgeleiteten Sicherheitsmaßnahmen wird den entsprechenden elementaren Gefährdungen entgegengewirkt. Die Buchstaben in der zweiten Spalte (C = Vertraulichkeit, I = Integrität, A = Verfügbarkeit) zeigen an, welche Grundwerte der Informationssicherheit durch die Anforderung vorrangig geschützt werden. Die folgenden elementaren Gefährdungen sind für den Baustein CON.8 *Software-Entwicklung* von Bedeutung.

G 0.14 Ausspähen von Informationen (Spionage)

G 0.15 Abhören

G 0.18 Fehlplanung oder fehlende Anpassung

G 0.19 Offenlegung schützenswerter Informationen

G 0.20 Informationen oder Produkte aus unzuverlässiger Quelle

G 0.22 Manipulation von Informationen

G 0.23 Unbefugtes Eindringen in IT-Systeme

G 0.25 Ausfall von Geräten oder Systemen

G 0.28 Software-Schwachstellen oder -Fehler

G 0.29 Verstoß gegen Gesetze oder Regelungen

G 0.30 Unberechtigte Nutzung oder Administration von Geräten und Systemen

G 0.31 Fehlerhafte Nutzung oder Administration von Geräten und Systemen

G 0.32 Missbrauch von Berechtigungen

G 0.39 Schadprogramme

G 0.40 Verhinderung von Diensten (Denial of Service)

G 0.41 Sabotage

G 0.46 Integritätsverlust schützenswerter Informationen