IT-Sicherheitsanforderungen

SICHERHEITSANFORDERUNGSMODELLIERUNG

Die digitale Sicherheit eines IT-Systems beginnt mit der Spezifikation der funktionalen und nichtfunktionalen Sicherheitsanforderungen. Anwender, die Wert auf die Sicherheit ihrer IT-Systeme legen, müssen schon bei deren Anforderungsanalyse die geeigneten Sicherheitsmaßnahmen vorsehen. Was nicht gefordert wird, muss auch nicht realisiert werden. Wer Sicherheit haben will, muss sie explizit fordern. Dieser Artikel schildert anhand praktischer Beispiele, wie Sicherheitsanforderungen zu formulieren und zu kontrollieren sind.

Maßnahmen wie Datenzugriffsschutz, statische Codeanalyse und dynamische Ablaufverfolgung sind typische Sicherheitsmaßnahmen, die vom Entwicklungsprojekt zu verlangen sind. Die fertigen Softwareprodukte dürften nur vom Kunden angenommen werden, wenn alle Anforderungen, einschließlich die Sicherheitsanforderungen, erfüllt sind, da mit der Software-Akzeptanz durch den Auftraggeber auch Garantien, Gewährleistungs- und teilweise hohe Haftungsansprüche verknüpft sind. Öffentliche Ämter und private Firmen, die aufgefordert werden, die digitale Sicherheit ihrer IT nachzuweisen, können ein technisches Sicherheits-audit beauftragen. Wenn sie die Prüfung, kurz das Audit, bestehen, erhalten sie ein Sicherheitszertifikat, welches die Übereinstimmung der Sicherheitsanforderung mit den implementierten Funktionen in der Software bescheinigt.

Zweck eines Sicherheitsaudits

Mittlerweile bieten gewerbliche oder staatlich beeidete IT-Unternehmen eine Sicherheitszertifizierung als Dienstleistung an. Allerdings müssen sie selbst von einer staatlichen Instanz akkreditiert werden. Wie beim TÜV oder bei den IT-Zivilingenieuren haben nur staatlich akkreditierte Prüfinstanzen das Recht, anerkannte Zertifikate auf Basis von Normen und Standards auszufertigen. In Österreich sind das vorzugsweise die IT-Zivilingenieure, die bereits zur Zeit des Kaisers Franz Joseph institutionalisiert wurden. Ihre Aufgabe besteht darin, Projekte und Anschaffungen im Namen des Staates zu kontrollieren. Zu diesen Anschaffungen zählen heute auch sicherheitsrelevante Softwaresysteme. Der Zivilingenieur prüft, ob ein gekauftes oder entwickeltes Softwareprodukt dem vereinbarten Leistungsumfang entspricht. Die Basis solcher Prüfungen ist die Anforderungsspezifikation beziehungsweise in der Welt der Behörden das Lasten- und das Pflichtenheft. Diese beiden Dokumente sind auch die Basis von Ausschreibungen an externe Auftragnehmer. Das Lastenheft beschreibt, was der Auftraggeber haben will. Das Pflichtenheft beschreibt, was der Auftragnehmer zu liefern „gedenkt“. Zusammen ergeben sie die Anforderungsdokumentation. Die Sicherheitsanforderungen sind dabei ein wesentlicher Bestandteil dieser beiden Dokumente. Das Ziel eines Sicherheitsaudits ist, zu bestätigen, dass die gesetzlich vorgeschriebenen Maßnahmen schriftlich angefordert und auch erfüllt werden. Niemand soll dem Anwender vorwerfen können, er habe die Sicherheit der Software und den Schutz der Daten vernachlässigt [BSI-2016].

Durchführung von Sicherheitsaudits

Ein Sicherheitsaudit ist die Voraussetzung für die Erteilung eines Sicherheitszertifikats für Softwaresysteme. Der Weg dazu führt über ein Zertifizierungsverfahren namens „CYBERBELT©“. Das Verfahren sieht sechs Auditschritte vor:

1. das Audit der Anforderungsdokumentation (Lasten-/Pflichtenheft),
2. die Ableitung eines Anforderungsmodells,
3. das Audit des Quelltexts (Sourcecodes),
4. die Ableitung eines Quelltext-Modells,
5. der Abgleich des Quelltext-Modells mit dem Anforderungsmodell,
6. das Audit der sicherheitsrelevanten Software im laufenden Betrieb.

Nur wenn die Software auf den drei Stufen: (1) Anforderungsdokumentation, (3) Quelltext und (6) Betrieb als ausreichend sicher bewertet wird, wird eine staatlich befugte und beeidete Zertifizierung auf Basis eines Ziviltechnikergutachtens ausgesprochen. Im vorliegenden Beitrag wird lediglich die erste Stufe – die Anforderungsdokumentation – als Basis für die weiteren Schritte vorgestellt. Sowohl das Audit des Quelltexts – Code-Audit – als auch des Systembetriebs sind in der Literatur zur Softwaresicherheit ausreichend beschrieben [VieMcGr00].

Audit der Anforderungsdokumentation

Im ersten Schritt werden die Anforderungsdokumente restrukturiert, markiert und gemessen. Bei der Restrukturierung wird der Gesamttext in kleinere Teiltexte zerlegt. Jeder Teiltext bildet einen eigenen Abschnitt. Bei der Markierung der sicherheitsrelevanten Anforderungen werden Schlüsselwörter in den Text eingearbeitet (tagging), um die Entitäten beziehungsweise Objekte im Text zu identifizieren. Die Schlüsselwörter stehen vor der Anforderung im (Ab-)Satz, in dem die Entitäten vorkommen.
Beim Audit der Anforderungen werden Regelverletzungen und Entitäten gezählt. Die Regelverletzungen werden in einem Anforderungsmängelbericht für die Qualitätssicherung zusammengefasst [Sneed07]. Die Entitätszahlen werden in einem Messbericht für die Systemplanung ausgegeben [Sneed18]. Die quantitative Messung ist nur möglich, wenn das Anforderungsdokument in Form eines Modells mit festdefinierten, zählbaren Entitäten vorliegt. Erst das Modell ermöglicht den Vergleich mit anderen Anforderungsdokumenten mit denselben Entitäten-Typen.

Ableitung eines Anforderungsmodells für die Sicherheit eines IT-Systems

Das Ziel der Sicherheitsmodellierung ist es, ein Anforderungsmodell für die Sicherheit eines Softwaresystems aus dem Text der Anforderungsdokumente abzuleiten. Eine ordentliche Anforderungsdokumentation beschreibt nicht nur die einzelnen Anforderungen, geteilt in funktionale und nichtfunktionale Anforderungen. Sie beschreibt auch etliche Merkmale des gewünschten Softwaresystems wie Ziele, Regeln, Anwendungsfälle, Datenobjekte und Schnittstellen [Briand17]. Es sollte möglich sein, aus den Anforderungsdokumenten ein Anforderungsmodell abzuleiten.

Das Anforderungsmodell dient wiederum als Basis für die algorithmische Aufwandschätzung sowie für den modellbasierten Test. Die Sicherheitsanforderungen, -regeln, -anwendungsfälle, -objekte, und -schnittstellen gehören alle dazu. Sie bilden das Sicherheitsmodell – ein Untermodell des Anforderungsmodells. Es ist nicht zu leugnen, dass ein solches Modell bezüglich der Anzahl Entitäten-Typen bald an seine Grenzen stoßt. Deshalb muss der Modellierer sich auf ein Teilmodell beschränken wie hier das Sicherheitsmodell. Aus dem Prosatext wird ein Modell abgeleitet und in einer objektrelationalen Datenbank abgelegt. Das Modell schildert die Beziehungen zwischen den Sicherheitsentitäten (siehe **Abbildung 1**).

Ermittlung der Sicherheitsanforderungen

Falls nach der Zertifizierung ein Sicherheitsvorfall auftritt, haftet für den entstandenen Schaden im Regelfall derjenige, der die Software zertifiziert hat, zum Beispiel der zuständige IT-Zivilingenieur. Deshalb ist es unerlässlich, das Sicherheitsaudit so professionell wie möglich durchführen zu lassen. Der Sicherheitsauditor muss sicher sein, dass alle möglichen Bedrohungen zum System erkannt und ihre Abwehr als Anforderung erfasst wird.

Es gibt mindestens drei Quellen für Sicherheitsanforderungen:

- die allgemeine Literatur über bekannte Bedrohungen für IT-Systeme,
- die eigene Erfahrung mit anderen Systemen,
- die Simulation der Systemnutzung.

Für die simulierte Systemnutzung gibt es zwei Ansätze den Penetrationtest- und den Red-Team-Operations-Ansatz. Beim Penetrationtest-Ansatz werden sämtliche Möglichkeiten erprobt, in das System von außen in einem Black-Box-Ansatz, ohne spezielle Vorkenntnisse der Software, einzubrechen. Das erfordert viele Testfälle und kann sehr lange dauern. Außerdem wird das firmeninterne Sicherheitsteam (Blue Team) im Regelfall vorher informiert, welche Angriffe geplant sind, und es kann sich darauf vorbereiten.
Mit dem Red-Team-Ansatz werden nur jene Einbruchsmöglichkeiten getestet, die für dieses System gefährlich sein könnten. Dieser Ansatz ist demzufolge flexibel und auf neue, unerwartete Situationen ausgerichtet. Die Systementwickler werden im Gegensatz zum Penetrationstest nicht vorgewarnt. Sie dürfen nicht wissen, was im Test auf das System zukommt. Die Auswahl der Angriffsfälle wird vom Ausmaß des Schadens bestimmt. Jene Fälle, die den größten Schaden verursachen können, haben die höchste Priorität [AbGrlb17]. Die Analyse der Anforderungen dient dazu, die Anforderungen von den Endbenutzern zu sammeln und auszuwerten, beziehungsweise welche Daten sind wie zu speichern und welche Funktionen sind mit diesen Daten wie auszuführen. Die Endbenutzer werden jedoch selten mit allen Sicherheits- und Datenschutzbestimmungen vertraut sein. Deshalb werden die Anforderungsanalytiker darüber hinaus weitere Anforderungen aus den geltenden Gesetzen und Betriebsbestimmungen übernehmen müssen. Das Ergebnis ist eine Liste aller erkennbaren Anforderungen an das geplante System.

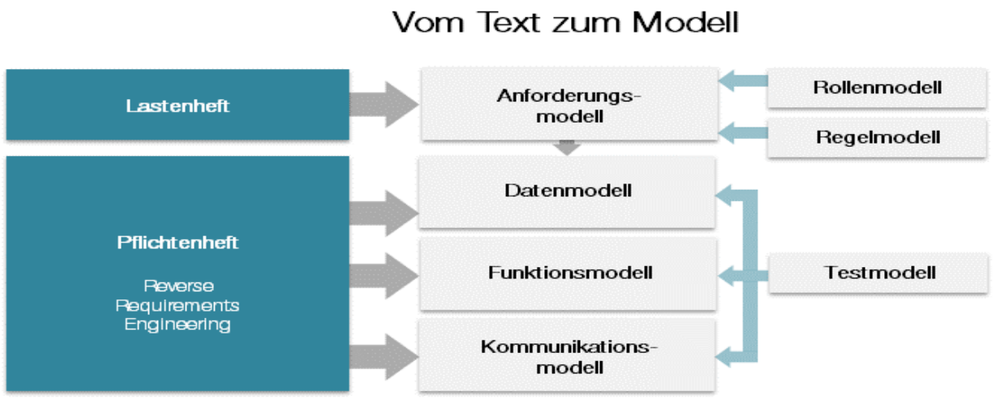

Abb. 1: Ableitung eines Anforderungsmodells

Sollte das System schon zumindest teilweise existieren, sind aus dem Code der implementierten Daten und Funktionen Anforderungen an die nächste Version durch Reverse-Engineering-Techniken zu gewinnen [Prywes96].

Spezifikation der Sicherheitsanforderungen

Sind die Sicherheitsanforderungen einmal erkannt, kann der Analytiker dazu übergehen, sie zu spezifizieren. Anforderungs-basiertes Testen geht davon aus, dass derjenige, der die Anforderungen schreibt, eine klare Vorstellung von dem hat, was das System leisten soll. Die Anforderungen drücken in Klartext aus, was die Benutzer vom System erwarten. Neben den funktionalen Anforderungen, die das „Was“ beschreiben, gibt es auch noch die nichtfunktionalen Anforderungen, die das „Wie“ beschreiben. Letztere werden auch als qualitative Anforderungen bezeichnet und geben vor, wie schnell, wie benutzerfreundlich, wie zuverlässig, wie effizient und auch wie sicher das System sein muss. Der Unterschied zwischen funktionalen und nichtfunktionalen Anforderungen ist nicht immer klar. Man könnte behaupten, die Authentifikation der Benutzer ist eine nichtfunktionale Anforderung, weil es der Sicherheit dient und nicht einer bestimmten Geschäftsfunktion, aber in der Tat ist dies eine Aktion, die im Quelltext zu finden ist. Entweder wird sie ausgeführt oder nicht. Eine nichtfunktionale Anforderung ist, wenn der Benutzer verlangt, dass mindestens 98 Prozent der Penetrationsversuche abgewehrt werden. Sie gilt für das System als Ganzes. Entweder wird sie erfüllt oder nicht erfüllt. Nichtfunktionale Anforderungen beziehen sich auf ein angestrebtes Qualitätsmerkmal der Software und sind meistens mit einer rationalen Metrik zu beantworten, wie in **Beispiel 1**.

```
&NFREQ_15 Abwehr_illegaler_Eintrittsversuche Metrik = NR_
ABGEWIESENER_VERSUCHE/NR_EINTRITTSVERSUCHE_EI-
NER_ UNAUTHORISIERTER_PERSON)    Muss   > 0.90 sein
```

Beispiel 1: Eine nichtfunktionale Sicherheitsanforderung

Hinter den Sicherheitsanforderungen stecken allgemeine Grundwerte. Laut der Richtlinie des Bundesamts für Sicherheit in der IT gibt es primär drei fundamentale Grundwerte: Vertraulichkeit, Integrität und Verfügbarkeit [BSI-2013]:

- Vertraulichkeit ist der Schutz vor unbefugter Preisgabe von Informationen. Das Belauschen, eine Weitergabe oder Veröffentlichung von vertraulichen Informationen ist daher nicht erwünscht.
- Die Integrität von Daten gewährleistet deren Vollständigkeit und Unversehrtheit über einen bestimmten Zeitraum und ermöglicht die Aufdeckung einer Manipulation.
- Verfügbarkeit kann sich auf IT-Systeme oder Daten beziehen und fordert, dass sie zu definierten Zeiten, im Einklang mit der Vertraulichkeit und der Integrität, von Anwendern stets wie vorgesehen genutzt werden können.

Die Grundwerte beziehen sich nicht ausschließlich auf Daten, sondern sollten ebenfalls für Datenflüsse, Kommunikationskanäle oder ganze IT-Systeme gelten.

- **Beispiel:** Der Auftragnehmer verwendet die jeweils dem aktuellen Stand der Technik entsprechenden Sicherheitstechnologien bei der Bereitstellung der Services.
- **Beispiel:** Der Auftragnehmer verpflichtet sich, die gültigen Sicherheits-Richtlinien und -Verfahren entsprechend zu beachten.
- **Beispiel:** Die Vertraulichkeit der Kunden muss gewährleistet sein. Nur vertrauenswürdige Kunden können das System nutzen.

Beispiel 2: Schwache Anforderungen

Es obliegt dem Anwender in Anlehnung an diese Grundwerte, seine Sicherheitsanforderungen zu ermitteln und so zu dokumentieren, dass das betreffende System gegen sie getestet werden kann.

Beispiele schwacher Sicherheitsanforderungen
Schwache Sicherheitsanforderungen sind solche, die schwammig formuliert sind und keine eindeutige Antwort herausfordern, ob sie erfüllt sind oder nicht, siehe **Beispiel 2**.
Die Erfüllung solcher Anforderungen lässt sich nicht eindeutig bestätigen. Sie sind zu abstrakt formuliert. Exakte Testfälle lassen sich nicht abzuleiten.

Beispiele starker Sicherheitsanforderungen
Starke Sicherheitsanforderungen sind solche, deren Erfüllung eindeutig mit „Ja“ oder „Nein“ zu bestätigen ist, oder sie sind mit einem numerischen Wert messbar, zum Beispiel: mindestens 60 Prozent aller Eingangsparameter werden geprüft (siehe **Beispiel 3**).

- **FREQ_03 Datensicherung:** Der Auftragnehmer wird die im Rahmen der vertragsgegen-ständlichen Leistungen gespeicherten Daten durch Backup2Disk oder Backup2Tape tur-nusmäßig wie folgt sichern: Stündliche Sicherung von Datenbanken mit einer Vorhaltezeit von einem Tag. Tägliche Vollsicherung aller Applikations- und Datenbankserver mit einer Vorhaltezeit von 14 Tagen. Eine Jahresvollsicherung des Vorjahres am ersten Arbeitstag des neuen Jahres mit einer Vorhaltezeit von einem Jahr.
- **FREQ_05 Benutzer_Autorisierung:** Nur wenn der Benutzer autorisiert ist, kann er Akti-onen auf Objekten ausführen, die ihm durch seine Rolle zugeordnet sind. Query over the method = AuthorizationFacade. checkAccessUser.
- **FREQ_07 Check_für_SQL_Injection:** In den Datenbankzugriffsaufrufen dürfen keine Original-SQL-Anweisungen vorkommen. Das wäre eine Einladung zur SQL-Injection.
- **FREQ_09 Ungültige_Parameter:** 75 Prozent aller eingehenden Nachrichten sind auf ungültige Parameter zu prüfen und notfalls mit einer Fehlermeldung abzuweisen.

Beispiel 3: Schwache Anforderungen

Sicherheitsanwendungsfälle

Funktionale Anforderungen werden durch Anwendungsfälle implementiert. Um als implementiert zu gelten, muss eine Anforderung zunächst einem Anwendungsfall zugewiesen sein. Darüber hinaus wird der Anwendungsfall einem oder mehreren Codeblöcken wie Funktionen in C, C++ und C#, Paragrafen in COBOL, Prozeduren in PL/I oder Methoden in Java zugeordnet werden. In der Anforderungsdokumentation ist es wichtig, für jede funktionale Anforderung einen oder mehrere Anwendungsfälle zu haben. Das hat zur Folge, dass manche Sicherheitsanforderungen durch einen eigenen Anwendungsfall und andere durch einzelne Schritte in verschiedenen anderen Anwendungsfällen erfüllt werden (siehe **Abbildung 2**).

Es obliegt dem Sicherheits-Analytiker, die Erfüllung der Sicherheitsanforderungen in den Anwendungsfällen einzubauen, zum Beispiel die Anforderung, die Benutzer eines bestimmten Anwendungsfalls zu autorisieren, oder die Anforderung, die Zugriffe auf eine bestimmte Datenbank zu kontrollieren. Die reinen Sicherheitsanwendungsfälle werden wie andere Anwendungsfälle einen Auslöser, Vor- und Nachbedingungen, einen oder mehrere Pfade mit jeweils n Schritten, eine oder mehrere Eingaben und Ausgaben, einen Vorgänger, einen Nachfolger und eine Ausnahmebedingung ausweisen (siehe **Beispiel 4**).

Definition von Sicherheitsprüfungen in anderen funktionalen Anwen-dungsfällen

Ein Großteil der Sicherheitsanforderungen wird in den anderen Anwendungsfällen als Schritte spezifiziert, zum Beispiel die Anforderung, alle Datenbankzugriffe auf SQL-Injection zu prüfen. Der Datenbankzugriff wird in einem Anwendungsfall wie „Update_CustomerData“ stattfinden. Dort sollte es einen Schritt in dem Zugriffspfad geben, der die SQL-Parameter prüft. Wenn nicht, ist die Anforderung nicht er-füllt. Beispiel 5 zeigt, wie dieser Anwendungsfall aufgebaut werden könnte.
der Sicherheitsanforderung ist in der Be-schreibung des Anwendungsfalls durch das Attribut „erfüllt“ explizit spezifiziert. Damit ist klargestellt, dass die Anforde-rung zu diesem Anwendungsfall zuge-wiesen ist. Sollte dieses Attribut fehlen, kann der Textparser den Text der Anfor-derung mit dem Text des Anwendungs-falls abgleichen und gemeinsame Begrif-fe suchen. Wenn annähernd die gleichen Begriffe im Anforderungstext und im Anwendungsfalltext er-

UC-01 User_Logon:
To open a session, the user must first be authorized.
Trigger: User_Login.
PreCondition: This user is not known to the system. Post-Condition: The user has a valid password and is accepted. Or the user has an invalid password and is rejected. Fulfills: FREQ_05 User Authorization.
MainPath_1:

1. User submits name.
2. User submits password up to three times (<=3).
3. If password is valid.
4. System checks user name in Table of authorized users.
5. If user Id is known to system the user is allowed to continue.

Inputs	Outputs
Table of authorized Users	Acknowledgement of Authori-zation or
Table of authorized Users	Rejection Message
User Id	
User Password (3)	

Beispiel 4: Ein Sicherheitsanwendungsfall

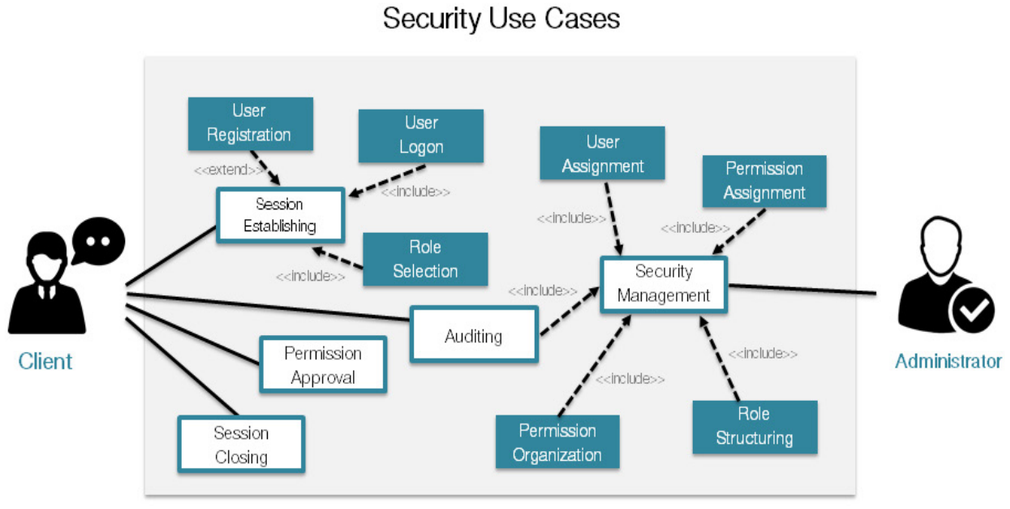

Abb. 2: Sicherheitsanwendungsfälle

scheinen, wird angenommen, dass die Anforderung zu dem Anwendungsfall passt und als „erfüllt“ gilt. Hier sind die gemeinsamen Begriffe: Customer_Name, Customer_Id, Custo-mer_Data, SQL_Injection und Security_ Violation.

Definition von Sicherheitsdatenobjekten

In dem Datenmodell der Anforderungsdokumentation werden gewisse Datenobjekte definiert, die nur zur Erfüllung der Sicherheitsanforderungen dienen, zum Beispiel die Tabelle der autorisierten Benutzer (siehe Beispiel 6). Die Sicherheitsobjekte werden von den Sicherheitsanwendungsfällen benutzt, um die Sicherheitsprüfungen durchzuführen. Sie werden einmalig erstellt und von dann an in jedem neuen Release aktualisiert. Sicherheitsobjekte unterscheiden sich von anderen Stammdatenobjekten, indem sie ausschließlich Daten beinhalten, die für die Sicherheitsmaßnahmen verwendet werden. Falls zusätzliche Sicherheitsmaßnahmen dem Zielsystem zugefügt werden, können die Sicherheitsdaten ohne Auswirkung auf die anderen Datenobjekte erweitert oder ergänzt werden.

Definition von Sicherheitstestfällen

Modellbasiertes Testen geht davon aus, dass die Testfälle aus einem Modell gewonnen werden [RBGW10]. Die Testfälle für einen Sicherheitstest werden aus dem Sicherheitsmodell gewonnen. Sie lassen sich automatisch aus dem deutschen Text ableiten, sofern die Pfade und Bedingungen erkennbar sind. Pfade sind mit einem Schlüsselwort zu markieren. Bedingungen werden mit einem Prädikat wie „if“, „falls“, „solange“ und „insofern“ begonnen. Der Text-Parser kann daran erkennen, dass dies eine bedingte Funktion ist. Der Satz wird ausgeschnitten und in eine Testfalltabelle eingefügt, wo er mit anderen Attributen ergänzt wird. Später können diese logischen Testfälle in physische Testfälle durch das Hinzufügen echter Datenwerte umgewandelt werden (siehe Beispiel 7).

UC-03 Update_CustomerData:
To update the customer data, the user must first gain access to the customer database. Trigger: User_selects_Update_Transaction.
PreCondition: The customer database contains the current data of all valid customers. PostCondition: The data of the selected customer is no longer the same as it was. Fulfills:
FREQ-04 Updating_Customer_Data
FREQ_07 Checking for_SQL_Injection.
MainPath_031:

1. User submits Customer_name and Customer_Id.
2. System checks parameters of database access call. <-
3. If access parameters are vulnerable to SQL Injection system reports security violation.
4. Otherwise customer data record is retrieved and data displayed.
5. If customer data is displayed user changes selected data fields.
6. Changed customer record is rewritten.

Erfüllt: FREQ_07 Checking_for_ SQL_Injection Exception Handling:

- Terminate Use case
- Display Rejection Message

Beispiel 5: Ein Standard-Anwendungsfall mit einer Sicherheitsfunktion

&Objekt_07 Authorized_Users = Table of authorized users contains following attributes:

- Authorized_User_Id (8-character Code)
- Authorized_User_Name
- Authorized_User_Address
- Authorizing_Office_Id (8-stelliger Code)
- Authorizing_Office_Name
- Date_of_Authorization (YY/MM/TT)
- Type_of_Authorization (limited,unlimited)
- Duration_of_Authorization (in Months)

Beispiel 6: Ein Sicherheitsdatenobjekt

Literatur & Links

[Briand17] L. Briand, Analyzing Natural Language Requirements – The not-too-sexy and yet curiously difficult Research that Industry needs, in: 23rd Int. Working Conf. on Requirements Eng., Foundation for Software Quality (REFSQ 17), 2017
[BSI-2013] Bundesamt für Sicherheit in der Informationstechnik: Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen an die Auftragnehmer, Bonn, 2013
[BSI-2016] BSI TR-03153 Technische Sicherheitseinrichtung für elektronische Aufzeichnungssysteme in der BRD, Bonn, 2016
[PottMcGr04] B. Potter, G. McGraw, Software Security Testing – building Security in, in: IEEE Security & Privacy, Sept., 2004, p. 81
[Prywes96] N. Prywes, Recovering Design and Specifications from Source Code, in: IEEE Software, Nov. 1996
[RBGW10] T. Roßner, C. Brandes, H. Götz, M. Winter, Basiswissen Modellbasierter Test, dpunkt.verlag, 2010
[Sneed07] H. Sneed, Testing against natural language Requirements, in: 7th IEEE Int. Conf. on Software Quality (QSIC2007), Portland, Oct. 2007, p. 380
[Sneed18] H. Sneed, W. Prentner, Requirements Reengineering, in: OBJEKTspektrum, 01/2019
[VieMcGr00] J. Viega, G. McGraw, E. Felten, Statically Scanning Java Code – Finding Security Vulnerabilities, in: IEEE Software Magazine, Sept. 2000, p. 68
[VieMcGr02] J. Viega, G. McGraw, Building secure Software – How to avoid Security Problems, Addison-Wesley, 2002

Test Case	TargetReq	Priority	TC-Type	TestCase Objective	Target Function	Objects
13	FREQ_05	High	Condition	If authorized, a user can perform actions on objects assigned to him by his role Query over the method = AuthorizationFacade.checkAccessUser	UC-02: Permission Approval	UserTab
14	FREQ_05	High	Condition	If not authorized, a user cannot perform actions on objects not assigned to him by his role. Query over method == AuthorizationFacade.checkAccessUser	UC-02: Permission Approval	UserTab
22	FREW_06	Med	DB	To test if the method AuthorizationFacade.getObject returns an authorized user	UC-03: Access User Record	PatientDB

Beispiel 7: Sicherheitstestfälle

Schlussfolgerung

Die Sicherheit von verteilter Software, speziell im Internet und in der Cloud, hängt von den Sicherheitsanforderungen ab. Was nicht angefordert wird, muss auch nicht getestet werden. Es obliegt dem Software-Sicherheits-Tester zu entscheiden, was er testet und was nicht. Wenn die sicherheitsrelevanten Anforderungen nicht vollständig, eindeutig und konsistent spezifiziert sind, wird die Software seine Sicherheitsanforderungen nicht erfüllen. Alle Sicherheitsverletzungen müssten erkannt und Anforderungen formuliert werden, um sie frühzeitig zu verhindern. Was nicht explizit gefordert wird, muss auch nicht implementiert werden.

Wenn Zeitdruck zur maßgeblichen Größe wird, werden im Regelfall als Erstes die Sicherheitsmaßnahmen reduziert oder ganz geopfert, da es anfangs keiner so schnell bemerkt. Ein Sicherheitsaudit soll das aufdecken. Dazu gehört die Prüfung, Messung und Modellierung der Sicherheitsanforderungen. Fehlende Sicherheitsmaßnahmen werden dadurch erkannt und ausgewiesen. Durch einen Abgleich des Sicherheits-Anforderungsmodells mit dem Quelltextmodell werden außerdem die nicht implementierten Sicherheitsmaßnahmen aufgedeckt. Ziel ist es, so viele Sicherheitsmängel wie möglich vor dem eigentlichen Akzeptanztest zu finden. Sicherheitsvorfälle können, speziell im Lichte der europäischen Datenschutzgrundverordnung, sehr teuer und auch nicht zuverlässig vermieden werden, wenn die Anforderungen und die Testfälle unvollständig sind.

DIE AUTOREN

DI Dr.tech Wolfgang Prentner
IT-Ziviltechniker . Informatiker . CEO ZTP.digital
prentner@ztp.digital . www.ztp.digital

ist seit 1998 IT-Ziviltechniker im Fachbereich Informationstechnologie. Außerdem ist er Geschäftsführer der ZTP.digital, Gerichtssachverständiger und promovierter Informatiker an der TU Wien. Als unabhängige Prüf- und Überwachungsstelle für Informatik, CyberSecurity, Datenschutz und dem CYBERBELT unterstützt Dr. Prentner in ehrenamtlicher Funktion die Länderkammer, die Bundeskammer und das Bundeskomitee Die Freien Berufe Österreichs sowie das österreichische Bundeskanzleramt seit 2004.

Harry M. Sneed
Consultant
harry.sneed@t-online.de . www.ztp.digital

ist zurzeit Software-Tester, -Vermesser und -Reengineer bei ZTP.digital in Wien. Nebenbei lehrt er Software-Rengineering an der TU Dresden sowie Software-Evolution und Softwaremessung an den Fachhochschulen Hagenberg und Wien. Er erhielt den Stevens Award von der IEEE 2009 für seinen Beitrag zur Software-Maintenance, den ersten Preis für Qualitätssicherung von der ASQF 2011 und die Auszeichnung als internationaler Tester des Jahres von der ISTQB im Jahre 2013.