

# Java Web Security

Dominik Schadow

Sichere Webanwendungen
mit Java entwickeln

dpunkt.verlag

**Dominik Schadow** arbeitet als Senior Consultant beim IT-Beratungsunternehmen bridgingIT und unterstützt Kunden in unterschiedlichen Projekten u.a. bei der Entwicklung von sicheren Java-Webanwendungen. Er ist Sprecher auf verschiedenen Konferenzen rund um die Themen Java und sichere Softwareentwicklung. In seiner Freizeit leitet er das Open-Source-Projekt JCrypTool, mit dem Anwender für die Kryptografie begeistert werden sollen und gleichzeitig ihre eigenen Krypto-Plug-ins entwickeln können.

**Dominik Schadow**

# Java-Web-Security

## Sichere Webanwendungen mit Java entwickeln

dpunkt.verlag

Dominik Schadow
info@dominikschadow.de

Lektorat: René Schönfeldt
Copy-Editing: Friederike Daenecke, Zülpich
Satz: Da-TeX, Leipzig
Herstellung: Frank Heidt
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Buch 978-3-86490-146-1
PDF 978-3-86491-448-5
ePub 978-3-86491-449-2

1. Auflage

Wieblinger Weg 17
691123 Heidelberg



5 4 3 2 1 0

# Inhaltsverzeichnis

# 1 Einleitung

Die Entwicklung von sicheren Webapplikationen ist ein wichtiges und herausforderndes Thema für jeden Java-Entwickler. Vielfältige Aufruf- und Verwendungsmöglichkeiten von Java-Core-Funktionalität und Frameworks, komplexe individuelle Webanwendungen mit zahlreichen angebundenen Systemen und kreative Angreifer sind nur einige der in diesem Umfeld zu bewältigenden Aufgaben. Vor dem Hintergrund von Java-Anwendungen in der Cloud, serviceorientierten Architekturen und auch ganz »normalen« Enterprise-Webanwendungen bleibt die Webapplikationssicherheit – unabhängig von aktuellen Hypes – eine der wesentlichen Herausforderungen der nächsten Jahre.

Zweifellos hat man allgemein erkannt, dass wir simple sicherheitsrelevante Programmierfehler und dadurch unsichere Webapplikationen heute nicht mehr länger tolerieren können. Immer mehr gespeicherte (Kunden-)Daten bedeuten schließlich auch, dass bei einem erfolgreichen Angriff sehr viel mehr Daten verloren gehen können. Gleichzeitig werden Angriffe immer standardisierter durchgeführt. Als Angreifer kommen so neben »professionellen« Übeltätern gleichermaßen Amateure mit nur geringen Entwickler- und IT-Kenntnissen infrage. Angriffe auf Webanwendungen können so zumindest teilweise per simplem Copy & Paste aus fertigen »Angriffs-Kits« durchgeführt werden.

Viele dieser Angriffe können Sie bereits im Vorfeld durch eine sicherheitsorientierte Softwareentwicklung verhindern oder zumindest erschweren. Doch was bedeutet die Entwicklung von sicherer Software – speziell von sicheren Webanwendungen – konkret? Was müssen Sie bei der Entwicklung einer Java-Webanwendung beachten, um häufig gemachte Fehler und die größten Risiken zu umgehen? Und welche Gefahren sind das eigentlich genau? Diese und viele weitere Fragen rund um die sichere Softwareentwicklung werde ich in diesem Buch beantworten, und ich werde Ihnen zeigen, wie einfach Ihre ersten Schritte in der Welt der sicheren Entwicklung von Java-Webapplikationen sein können.

## 1.1 Über dieses Buch

Das Know-how der Entwickler macht den Unterschied zwischen sicherer und unsicherer Software aus. Ihre tagtäglichen Entscheidungen bestimmen, ob eine Webanwendung zuverlässig und sicher ihren Dienst verrichtet oder ob sie aufgrund von Sicherheitslöchern traurige Berühmtheit erlangt. Auf welcher Seite dieser beiden Extreme sich Ihre Webanwendung wiederfindet, bestimmen Sie vor allem durch drei Dinge: durch eine gute Ausbildung der Entwickler im Umfeld der Softwaresicherheit (u. a. zur Schaffung des notwendigen Sicherheitsbewusstseins), durch eine nach diesen Richtlinien durchgeführte sichere Entwicklung sowie durch Sicherheitstests während und nach der Entwicklung der Software.

Allerdings ist Sicherheit, egal ob IT-Sicherheit im Allgemeinen oder sichere Softwareentwicklung im Speziellen, leider allzu häufig als trocken und wenig spannend verrufen. Viel zu wenige Entwickler beschäftigen sich intensiver mit diesem überaus wichtigen Thema. Gleichzeitig sind wir alle heutzutage von sehr vielen, mitunter auch täglich verwendeten Webapplikationen umgeben. Laufend kommen neue Webanwendungen hinzu. Immer mehr unserer Daten, darunter zunehmend solche sensibler Natur, sind in Webapplikationen erfasst. Die Sicherheit von Webanwendungen – die Sicherheit unserer Daten – wird damit zur wichtigsten Aufgabe aller an der Softwareentwicklung beteiligten Personen.

Mit dem vorliegenden Buch möchte ich Sie für die sichere Softwareentwicklung von Java-Webanwendungen begeistern und Ihnen zeigen, dass das Entwickeln sicherer Webanwendungen ebenso interessant und für jeden möglich sein kann wie die Entwicklung gewöhnlicher Software. Den ersten Schritt haben Sie bereits durch den Erwerb dieses Buches getan. Als Nächstes will ich Ihnen Ihre ersten praktischen Entwicklungsaktivitäten erleichtern und Ihnen zeigen, wie Sie als Java-Entwickler, notfalls unabhängig von anderen, in Ihrer täglichen Arbeit sichere Webanwendungen entwickeln können.

Sichere Software und die sichere Softwareentwicklung sind jedoch nahezu unerschöpfliche Themengebiete. Allein über die Hitliste der zehn kritischsten Websicherheitslücken – die sogenannten *OWASP Top 10* (Abschnitt 3.5.1) – ließe sich ein umfangreiches Buch schreiben, ganz zu schweigen von den *CWE/SANS Top 25* (einer weiteren (Un)Sicherheitshitliste (Abschnitt 3.5.2)). Alle darin aufgeführten Sicherheitsprobleme und -risiken sind prinzipiell wichtig und die meisten sind weit verbreitet. Allerdings zielen diese Listen, zumindest in einigen wenigen Punkten, eher auf Nicht-(Java-)Entwickler. Nicht alle Punkte sind daher für Entwickler gleichermaßen wichtig und interessant.

Deshalb habe ich dieses Buch geschrieben. Es konzentriert sich auf die Punkte, die meiner Erfahrung nach in der täglichen Entwicklungsarbeit häufig falsch gemacht oder schlichtweg vergessen werden. Ich stelle Ihnen dabei wo immer möglich praktisch einsetzbare und konkrete Gegenmaßnahmen vor, die Sie auch allein und nur in Ihrem eigenen Code verwenden können. Klar, niemand kann die Sicherheit einer im Team entwickelten Software allein gewährleisten, aber zumindest den Grundstein können Sie ohne weitere Unterstützung selbst legen.

Das Buch orientiert sich für Ihren Einstieg zunächst an den drei meiner Ansicht nach drängendsten Problemen bei Webanwendungen: Injections (Kapitel 6), Cross-Site Scripting (Kapitel 7) und Cross-Site Request Forgery (Kapitel 8) – definitiv drei der gefährlichsten und gleichzeitig verbreitetsten Sicherheitsprobleme. Aber natürlich sind dies bei Weitem nicht alle. Gleichzeitig können Sicherheitsprobleme nicht völlig isoliert voneinander betrachtet werden. Häufig öffnet eine Angriffsvariante erst die Tür für den folgenden schwerwiegenderen Angriff, oder ein Angriff wird erst durch das gleichzeitige Ausnutzen mehrerer Schwachstellen möglich. Neben diesen drei Hauptproblemen und den Gegenmaßnahmen finden Sie daher noch verschiedene dazugehörige kleinere Themen, die diese drei Angriffsformen unterstützen oder zumindest begünstigen. Auf diese Art erhalten Sie einen breiten Einblick in die sicherheitsorientierte Entwicklung von Java-Webanwendungen. Der Entwicklung von Webapplikationen, die vor den genannten Bedrohungen sicher sind, steht damit nichts mehr im Weg.

## 1.2 Zielgruppe und Voraussetzungen

Das vorliegende Buch richtet sich an Java-Entwickler, die bereits über umfangreiche Erfahrung mit der *Java Standard Edition* (Java SE) und zumindest mit den verbreiteten Teilen der *Java Enterprise Edition* (Java EE) verfügen. Zur Java EE zählen hierbei *JavaServer Pages*, *JavaServer Faces* und *Servlets*. Sie müssen kein Experte in der Java-Enterprise-Entwicklung und der Entwicklung von Java-Webapplikationen sein, sollten aber zumindest über ein fundiertes allgemeines Java-Know-how und Programmiererfahrung verfügen und eventuell kurz vor der Entwicklung Ihrer ersten großen Webanwendung stehen. Ganz ohne Kenntnisse im Umfeld von Java-Webanwendungen werden Sie an einigen Stellen Verständnisschwierigkeiten haben. Dagegen sind keine Kenntnisse in der IT-Sicherheit oder gar der Kryptografie notwendig, umgekehrt schaden diese aber auch nicht.

An den wenigen Abschnitten bisher haben Sie wahrscheinlich bereits bemerkt, dass ich im Text durchgehend männliche Bezeichnungen

verwende. Eine alle Geschlechter gleichermaßen ansprechende Variante – etwa Entwickelnde und Angreifende oder EntwicklerInnen und AngreiferInnen – ist beim Lesen nach einiger Zeit einfach nur noch anstrengend. Und als Entwickler respektive Entwicklerin sucht man ja gern den einfachsten Weg. Ich habe mich daher aus Gründen der Lesbarkeit durchgehend für die männliche Form entschieden. Ich hoffe, dass Sie, liebe Leserin und Entwicklerin, sich dadurch nicht weniger angesprochen fühlen und sich schon gar nicht vom Lesen des Buches abhalten lassen.

## 1.3 Webanwendungen

Bereits einige Male war nun schon die Rede von Webanwendungen bzw. Webapplikationen.[1] Auch wenn dieser Begriff nicht ganz so nebulös wie der einer (Java-)Enterprise-Anwendung ist, bedarf er doch einer kurzen Erläuterung zum besseren Verständnis im weiteren Buch. So sind bei Webanwendungen durchaus einige Unterschiede vorhanden, beispielsweise, ob die Webanwendung über eine eigene Weboberfläche verfügt oder »nur« über die Protokolle des Webs kommuniziert.

Viele Angriffe, darunter einige der hier im Buch beschriebenen, sind nur auf Webapplikationen möglich, die über eine eigene GUI verfügen – aber längst nicht alle. Angriffe über Injections oder Cross-Site Request Forgery beispielsweise funktionieren ebenfalls ohne grafische Oberfläche, z. B. über Webservices. Selbst wenn Angriffe unter diesen Bedingungen schwieriger werden, manchmal sogar unwahrscheinlich oder unmöglich scheinen mögen, sollten Sie sich bei nicht vorhandener grafischer Web-Benutzeroberfläche nicht zu sehr in Sicherheit wiegen. Sichere Softwareentwicklung ist grundsätzlich unabhängig davon, ob ein Endbenutzer bzw. ein Angreifer eine GUI sieht oder nicht.

Auch wenn ich in diesem Buch zum einfacheren Verständnis und zur leichteren Nachvollziehbarkeit anhand der Beispielanwendungen einen Angriff über eine Weboberfläche beschreibe, sollten Sie sich der Gefahren bei Webanwendungen ohne eigene GUI bewusst sein. Der Begriff Webanwendung bezeichnet in diesem Buch so allgemein wie möglich solche Anwendungen, die normalerweise über einen Browser aufgerufen und bedient werden – und die selbstverständlich überwiegend in Java entwickelt wurden. Ob diese Webanwendung aus mehreren Seiten oder nur einer einzigen besteht (Single-Page-Application), im Internet oder Intranet zugänglich ist, nur von angemeldeten Benutzern oder anonym verwendet werden kann, spielt, bis auf wenige Ausnahmen, keine Rolle.

[1] Im Buch verwende ich beide Begriffe synonym.

## 1.4 Abgrenzung

Trotz der im Buch angesprochenen Themen und Komponenten der Java Enterprise Edition ist dies kein vollständiges und allumfassendes Buch über die Sicherheit von Java-Enterprise-Applikationen. Bei der Entwicklung mit der *Java Micro Edition* (Java ME) profitieren Sie vermutlich ebenso vom ein oder anderen Tipp, allerdings steht diese Java-Edition nicht im Fokus. Sicherlich verhindern einige der vorgestellten Gegenmaßnahmen ähnliche oder gar identische Sicherheitsprobleme auf Mobilgeräten oder anderen eingeschränkten Geräten, allerdings existieren in diesen Umgebungen noch ganz andere Herausforderungen, die in diesem Buch keine Erwähnung finden.

Natürlich können nicht alle möglichen Sicherheitsprobleme rund um Java-Webapplikationen in diesem Buch abschließend behandelt werden. Selbst nachdem Sie Ihre Webapplikation gegen Injections, Cross-Site Scripting und Cross-Site Request Forgery abgesichert haben, ist Ihre Webapplikation nicht vor allen Angriffen und allen Angreifern sicher. Dafür sind die denkbaren Webanwendungen zu vielseitig und die dadurch möglichen Angriffe leider ebenfalls.

Gleichzeitig haben die im Buch vorgestellten Lösungen mit hoher Wahrscheinlichkeit keinen Bestand für alle Ewigkeit. Es ist ein Stück weit wie mit kryptografischen Algorithmen: Was heute sicher ist, kann morgen schon geknackt sein. Ganz so schlimm ist es bei der sicheren Softwareentwicklung zum Glück nicht: Die Grundlagen gelten in der Regel weiterhin. Die gezeigten Gegenmaßnahmen werden aller Voraussicht nach nicht von heute auf morgen vollständig wirkungslos. Allerdings liegt es durchaus im Bereich des Möglichen, dass ein kreativer Angreifer mit bestimmten Tricks einige der vorgestellten Gegenmaßnahmen umgehen kann. In diesem Fall werden dann Ergänzungen notwendig. Bleiben Sie daher auf dem Laufenden, folgen Sie den Java-Neuigkeiten im Internet, und passen Sie Ihre Webapplikationen bei Bedarf an.

Ganz im Sinne der Wiederverwendung werden im Buch verschiedene sicherheitsrelevante Open-Source-Frameworks vorgestellt, die ich bei der Entwicklung von sicheren Webanwendungen für hilfreich erachte. Die Frameworks selbst werden dabei nur kurz gestreift und in einigen Codebeispielen konkret in ihrer Anwendung gezeigt. Selbstverständlich stellt das keine umfassende Einführung in das jeweilige Framework dar. Manche Leser werden ihr Lieblingsframework im Buch auch ganz vermissen. Hier verändert sich das riesige Java-Universum einfach zu schnell, als dass ein allgemeines Buch zur sicheren Entwicklung mit Java jedes einzelne Framework in der gebotenen Tiefe behandeln könnte. Wichtiger als konkrete Framework-Kenntnisse sind

ohnehin die Java-Grundlagen zur sicheren Softwareentwicklung. Dennoch ist die Verwendung von Frameworks in jedem Fall sinnvoll und erleichtert gleichzeitig die Entwicklungsarbeit. Im Literaturverzeichnis finden Sie daher verschiedene weiterführende Bücher und in den einzelnen Kapiteln zahlreiche Links zu den angesprochenen Frameworks.

Noch eine kleine Einschränkung zum Schluss: Dies ist kein Hacker-Buch.[2] Natürlich stelle ich zum besseren Verständnis einige Angriffe auf Webapplikationen vor, allerdings nur so weit und so detailliert, wie dies für das Verständnis des Sicherheitsproblems notwendig ist. Die dabei vorgestellten und verwendeten Tools sind allesamt legal und können Sie bei Ihrer täglichen Arbeit als Entwickler unterstützen. Dass diese Tools teilweise ebenfalls von Angreifern verwendet werden, sollte uns Entwickler nicht von ihrer Verwendung abhalten. Als ehrliche und zuverlässige Entwickler testen wir ohnehin nur unsere eigenen Webapplikationen und führen keine (illegalen) Tests mit fremden Webapplikationen durch.

## 1.5 Der Quellcode zum Buch

Im Buch finden Sie zahlreiche und über die Kapitel hinweg voneinander unabhängige Beispiele mit Codeausschnitten. Den dazugehörigen vollständigen Quellcode in lauffähigen Webanwendungen finden Sie auf GitHub unter https://github.com/dschadow/Java-Web-Security. Sie können das vollständige Repository mit Git klonen oder als Archiv herunterladen. Alle Projekte stehen unter der *Apache Software License 2.0*.[3]

Sämtliche Projekte sind mit *Apache Maven*[4] angelegt und können in Ihre bevorzugte IDE importiert werden. Die Projekte sind nach Kapiteln benannt (z. B. `Ch06_SQLInjection`) und bauen nicht aufeinander auf (auch wenn Sie gewisse Gemeinsamkeiten entdecken werden). Zu umfangreicheren Kapiteln können mehrere Projekte vorhanden sein. Am Ende der Kapitel mit vorhandenen weiterführenden Projekten finden Sie einen Abschnitt namens *Beispielprojekte*, der auflistet, welche Projekte für dieses Kapitel relevant sind und was Sie in diesen Projekten ausprobieren und nachvollziehen können.

Die Beispielprojekte sind bewusst mit einem auf das jeweilige Sicherheitsproblem eingegrenzten Fokus angelegt. Das heißt beispielswei-

[2] Im Übrigen ist dies die einzige Stelle, an der ich negativ belastet von Hackern spreche. Im restlichen Buch verwende ich die korrekte Bezeichnung »Angreifer« für die Bösen.

[3] http://www.apache.org/licenses/LICENSE-2.0

[4] http://maven.apache.org

se, dass eine Datenbankverbindung zu einer In-Memory-DB nur verwendet wird, wo dies absolut notwendig ist. Andernfalls werden die Informationen per Java-Code zur Runtime generiert und verarbeitet und nach Beendigung der Anwendung wieder verworfen. Auf eine umfassende Gestaltung der GUI wurde bewusst verzichtet, um Ihnen im Code den Blick auf das Wesentliche zu erleichtern. Ausgaben werden dazu neben der Anzeige im Browser meist per simplem `System.out.println` oder `LOGGER` durchgeführt. Für die Ausführung der Projekte ist nur ein Webserver wie *Apache Tomcat*[5] notwendig. Sie können hierbei eine bereits auf Ihrem System vorhandene Installation verwenden oder das Beispielprojekt direkt über das per `Maven-Tomcat7-Plug-in` zur Verfügung gestellte `mvn tomcat7:run-war` starten. Umfangreichere Informationen zum Start jeder Webanwendung finden Sie immer in der `pom.xml` im jeweiligen Projekt, und allgemeine Hinweise stehen in der `Readme`-Datei des Repository.

Empfehlenswert ist die Verwendung von *Mozilla Firefox* und die Installation des *Firebug*-Add-ons zum Ausprobieren der Webanwendungen.[6]

Im GitHub-Repository-Wiki unter https://github.com/dschadow/Java-Web-Security/wiki finden Sie zusätzlich sämtliche Links aus dem Buch, geordnet nach Kapiteln. Das Abtippen der teils langen URLs bleibt Ihnen damit erspart.

## 1.6 Aufbau des Buches

In den folgenden beiden Kapiteln erhalten Sie zunächst zahlreiche allgemeine Informationen rund um die sichere Softwareentwicklung, bevor wir uns anschauen, wie es allgemein um die Sicherheit von Java bestellt ist. Die Kapitel 4 und 5 legen anschließend die notwendigen Grundlagen für die sichere Entwicklung von Java-Webanwendungen. Nach der Vermittlung dieser unbedingt erforderlichen Grundkenntnisse tauchen Sie direkt ein in die Welt von Injections, Cross-Site Scripting und Cross-Site Request Forgery. Die Kapitel 6, 7 und 8 können Sie daher in beliebiger Reihenfolge lesen. Allerdings sind viele der in den früheren Kapiteln vorgestellten Grundlagen zum Verständnis notwendig, vor allem die aus den Kapiteln 4 und 5. Abschließend stelle ich Ihnen in Kapitel 9 noch einige Tools vor, mit denen Sie sich die Arbeit bei der sicheren Softwareentwicklung deutlich vereinfachen und gleichzeitig Ihre Kenntnisse in der sicheren Softwareentwicklung erwei-

[5] http://tomcat.apache.org

[6] https://www.mozilla.org und https://addons.mozilla.org/de/firefox/addon/firebug

tern können. Ideen für Ihre weiteren Schritte in der Welt der sicheren Softwareentwicklung finden Sie in Kapitel 10.

Im Einzelnen behandeln die Kapitel folgende Themen:

**Kapitel 2:** Sicherheit von Anfang an

Eine sichere Webanwendung benötigt mehr als eine sichere Entwicklung. Dieses Kapitel zeigt, wo überall auf Sicherheit geachtet werden muss, welche Auswirkungen eine sichere Entwicklung auf die Entwicklungsdauer und -kosten einer Webanwendung hat und wie Altapplikationen abgesichert werden können.

**Kapitel 3:** Java ist doch schon sicher?!

Java hatte lange Zeit einen nahezu fantastischen Ruf in Bezug auf die Sicherheit. Das ist mit ein Grund dafür, dass sich Entwickler bisher nur wenige Gedanken über die Sicherheit gemacht haben. Welche Sicherheitsfeatures sind in Java aber tatsächlich vorhanden, und wie unterstützen sie die Sicherheit einer Webanwendung?

**Kapitel 4:** Java-Security-Basics

Dieses Kapitel vermittelt die notwendigen Grundkenntnisse für eine sichere Entwicklung mit Java. Es zeigt, wie Input-Validierung und Output-Escaping funktionieren und welche Frameworks Sie bei der Entwicklung unterstützen können. Auch die richtige Fehlerbehandlung in Webanwendungen wird vorgestellt.

**Kapitel 5:** Session-Management mit Java

Fehler beim Java-Session-Management können einige der in den folgenden Kapiteln vorgestellten Angriffe erst ermöglichen bzw. diese erleichtern. Umso wichtiger ist es, dass Sessions korrekt konfiguriert werden. Welche Probleme dabei auftreten können und wie Sie diese umgehen, erfahren Sie in diesem zweiten Grundlagenkapitel.

**Kapitel 6:** Injections

Injections befinden sich seit vielen Jahren auf Platz eins der Bedrohungen. Dieses Kapitel stellt neben der weithin bekannten und trotzdem noch immer weit verbreiteten SQL Injection mit der XPath Injection und der Log Injection zwei weniger bekannte Injections vor und zeigt, wie Sie diesen und anderen Injection-Angriffen wirkungsvoll begegnen können.

**Kapitel 7:** Cross-Site Scripting (XSS)

Cross-Site Scripting stellt ein weiteres sehr verbreitetes Sicherheitsproblem in Webanwendungen dar und dient oft als Ausgangsbasis für weitere Angriffe. Verschiedene Varianten machen Angriffe

dabei noch vielseitiger und Gegenmaßnahmen anspruchsvoller. In diesem Kapitel erfahren Sie, welche Gegenmaßnahmen tatsächlich wirksam sind und wie Sie diese kombiniert vor Cross-Site Scripting schützen können.

**Kapitel 8:** Cross-Site Request Forgery (CSRF)
Cross-Site Request Forgery hat das Potenzial, eigentlich im Intranet geschützte Webanwendungen anzugreifen und heimlich Operationen im Namen ordnungsgemäß angemeldeter Benutzer auszuführen. Die Gegenmaßnahmen zeigen Ihnen, wie Sie Ihre Webanwendungen unabhängig vom Einsatzgebiet absichern können.

**Kapitel 9:** Tools
Nachdem Sie in den vorangegangenen Kapiteln gelernt haben, bestimmten Bedrohungen zu begegnen, möchte ich Ihnen in diesem Kapitel noch einige hilfreiche Tools vorstellen, die Sie bei Ihrer täglichen Entwicklungsarbeit unterstützen. Neben der Codeanalyse zeige ich Ihnen verschiedene hilfreiche Anwendungen, die Sie beim Vertiefen Ihrer Sicherheitskenntnisse unterstützen können.

**Kapitel 10:** Ausblick
Am Ende des Buches sind Sie auf dem richtigen Weg zur sicheren Entwicklung von Java-basierten Webanwendungen. Aber selbstredend gibt es noch weitere Themen, die Sie vor, während und nach Ihrer Entwicklungstätigkeit beachten können. Dieses letzte Kapitel liefert Ihnen einige Anregungen.

## 1.7 Danksagungen

Wohl kaum ein Buch lässt sich allein schreiben, zahlreiche Unterstützer und Sparringspartner sind dazu notwendig. Das war auch beim vorliegenden Buch nicht anders. Sehr großer Dank gebührt meinem Lektor René Schönfeldt vom dpunkt.verlag, mit dem die Idee zu diesem Buch beim Java Forum Stuttgart 2012 entstanden ist.

Besonders danken möchte ich weiterhin meinen unermüdlichen Reviewern Yves Geissbühler, Frank Numrich, Christian Unglaube und Wolfgang Werner sowie zahlreichen weiteren Kollegen bei der bridgingIT fürs viele Lesen, Kommentieren und Diskutieren.

# 2 Sicherheit von Anfang an

*Die nichtfunktionale Anforderung »Sicherheit« stand und steht meiner Erfahrung nach häufig sehr weit unten am Ende jeder Prioritäten- und Wunschliste. Ein Grund dafür ist, dass man als Benutzer bzw. Auftraggeber normalerweise kaum etwas von der Sicherheit einer Anwendung mitbekommt, gefühlt also keine Gegenleistung für sein Geld erhält. Eine Investition in die Applikationssicherheit zahlt sich, wenn überhaupt, erst langfristig aus. Solange nichts passiert, war die Investition dagegen überhaupt nicht notwendig. Mit viel Glück kann man so selbst eine unsichere Webanwendung bis an ihr Lebensende betreiben, ohne dass es je zu einem Zwischenfall kommt. Durch die zunehmende Vernetzung von Webanwendungen werden Sicherheitsvorfälle allerdings immer wahrscheinlicher. Auf das Glück allein sollten Sie sich daher nicht mehr verlassen.*

## 2.1 Forderung nach Sicherheit

Die wenigsten Auftraggeber betrachten Sicherheit bzw. das Fehlen derselben bisher als *Showstopper*. Hier zeichnet sich zwar langsam ein Sinneswandel ab, allerdings wird noch immer kaum jemand »nur« wegen möglicher Sicherheitsprobleme eine verspätete Auslieferung einer neuen Webapplikation riskieren. Gleichzeitig fordert kaum ein Benutzer oder Auftraggeber Sicherheit explizit ein. Offensichtlich wichtiger sind bei der Entwicklung einer neuen Webanwendung fast immer neue Features, die Unterstützung neuer Geschäftsprozesse oder aber ein besseres und moderneres User Interface. Weiterhin ist das notwendige Wissen rund um die sichere Entwicklung in Entwicklerkreisen zumindest derzeit noch immer wenig verbreitet. Das sind nur einige der Gründe, die anschließend unsichere Webanwendungen hervorbringen.

*Sicherheit als nichtfunktionale Anforderung*

Häufig verbirgt sich der Wunsch nach Sicherheit im Lastenheft[1] als ein nicht näher spezifizierter Punkt in den nichtfunktionalen Anforde-

[1] Wikipedia erklärt die Eigenschaften und Unterschiede von Lastenheft (https://de.wikipedia.org/wiki/Lastenheft) und Pflichtenheft (https://de.wikipedia.org/wiki/Pflichtenheft).

rungen. Zwischen den Stichwörtern *Performance* und *Erweiterbarkeit* findet sich dort ein Punkt *Sicherheit*. Einzig aufgrund dieser kurzen Angabe gehen viele Auftraggeber davon aus, dass die entwickelte Software schon den üblichen Sicherheitserfordernissen genügen wird. Welche Sicherheitserfordernisse genau für die zu entwickelnde Software und die darin verfügbaren Daten vorliegen, wird dann viel zu selten exakt festgelegt und dokumentiert. Der Auftragnehmer führt seinerseits diesen Punkt mit kaum mehr Details im Pflichtenheft auf. Ob und wie diese Forderung nach Sicherheit bei der folgenden Entwicklung konkret beachtet wurde, erfährt und hinterfragt (testet) ein Auftraggeber im weiteren Verlauf häufig nicht. In der Regel tut er dies jedenfalls nicht, bis es zu spät ist.

Natürlich müssen Sie sich als Entwickler auf die explizit geforderten Features konzentrieren. Schließlich wollen sowohl Sie selbst als auch Ihr Unternehmen mit der Entwicklung der Webanwendung Geld verdienen. Einiges können und sollten Sie aber auch ohne die explizite Unterstützung von anderen, einschließlich des Auftraggebers, für eine sicherere Webanwendung unternehmen. Und bei Punkten, bei denen Sie Unterstützung benötigen, können Sie den Kunden in die richtige Richtung lenken und ihn auf die Wichtigkeit von sicherer Software hinweisen. Was genau Sie dazu selbst tun können, stelle ich Ihnen im weiteren Verlauf des Buchs vor.

*Hack yourself first*

Von Bedeutung ist in diesem Zusammenhang ebenfalls, dass Sie bei der Umsetzung nicht nur an die Funktionalität und das Entwickeln derselben denken, sondern sich zusätzlich Gedanken darüber machen, wie ein Angreifer die neuen und bereits existierenden Funktionen böswillig ausnutzen könnte.[2] Angreifer werden genau das tun. Um sie wirkungsvoll von einem erfolgreichen Angriff abhalten zu können, müssen Sie daher ein Stück weit genauso denken und Ihre eigene Webanwendung bereits vor den Angreifern selbst angreifen und anschließend weiter absichern. Details dazu finden Sie in Kapitel 9.

[2] Hier spricht man oft von »hack yourself first«, bekannt durch Jeremiah Grossman (http://tedxtalks.ted.com/video/TEDxMaui-Jeremiah-Grossman-Hack) und Troy Hunt (http://www.troyhunt.com/2013/05/hack-yourself-first-how-to-go-on.html).

**Hinweis: Sicherheit kommunizieren**

Warum wird die Sicherheit einer Anwendung meist nur nachlässig eingefordert? Eine Ursache dafür ist sicherlich, dass Personen außerhalb der Softwareentwicklung nur wenig bis gar nichts mit Begriffen wie Cross-Site Scripting oder Cross-Site Request Forgery anfangen können. Dazu kommt, dass Auftraggeber sich eigentlich auch nicht auf dieser Detailebene mit ihrem Bedürfnis nach Sicherheit beschäftigen wollen. Auf der anderen Seite ist verständlicherweise eine gewisse Erwartungshaltung gegenüber der neuen Software vorhanden. Die optimale Sicherheit der Webanwendung und von deren Daten gehören zweifelsfrei dazu. Sofern Sie daher Software sicher entwickeln oder gar einem *Security Development Lifecycle* (Abschnitt 2.4) folgen, sollten Sie dies explizit(er) in Ihren Pflichtenheften und Angeboten hervorheben und sich so von anderen Angeboten deutlicher unterscheiden. Auf diese Art werden Nichtentwickler und vor allem Auftraggeber ebenfalls für das Thema Sicherheit sensibilisiert.

## 2.2 Warum ist sichere Software wichtig?

Im laufenden Betrieb auftretende Sicherheitsprobleme lassen sich bis auf wenige Ausnahmen kaum kurzfristig und ohne Auswirkungen auf andere Bereiche der Webanwendung beheben. Derartige Probleme erfordern fast immer umfangreichere und komplexere Korrekturen an vielen verschiedenen Punkten in der Software und über mehrere Anwendungsschichten hinweg. Diese Korrekturen lassen sich darum nicht über Nacht einfach und sicher umsetzen.

Das viel größere Problem ist aber, dass auch der schönste Security-Bug-Fix einmal gestohlene Daten nicht wieder zurückbringt. Oder den ruinierten Ruf eines Unternehmens so einfach wiederherstellt. Oder die über die eigene Webapplikation ausgeführten Angriffe auf andere Webapplikationen wieder rückgängig macht. Deswegen muss sichere Software die in ihr verfügbare Funktionalität und die in ihr verarbeiteten Daten von Anfang an schützen.

*Sicherheit von Anfang an*

Im Umfeld der sicheren Softwareentwicklung folgt daraus, dass eine Anwendung, vor allem eine exponierte Webanwendung, vom ersten Moment ihrer Inbetriebnahme an sicher sein muss. Die für eine Webanwendung und deren Daten zusätzliche Sicherheit hängt dabei erst ab einem gewissen Punkt von der Wichtigkeit der Webanwendung und der Kritikalität ihrer Daten ab. Einen gewissen Grundschutz benötigen alle Webapplikationen. Dieser Grundschutz beinhaltet den Schutz vor bekannten Schwachstellen und Programmierfehlern, wie sie in den folgen-

den Kapiteln vorgestellt werden. Der darüber hinausgehende Schutzbedarf ist individuell verschieden, ebenso wie die Maßnahmen, mit denen Sie für ausreichenden Schutz sorgen, wie beispielsweise eine *Web Application Firewall* (Abschnitt 2.7.2). Generell benötigen wir heutzutage aber mehr sichere Software und nicht mehr Sicherheitssoftware.

Den angesprochenen Grundschutz müssen Sie daher an jeder von außen erreichbaren Schnittstelle (z. B. einer Benutzeroberfläche oder an einem Webservice) Ihrer Webanwendung gewährleisten, und zwar gegen jeden denkbaren Angriff. Ein unfaires Spiel, schließlich genügt einem Angreifer oft eine einzige ungesicherte bzw. verwundbare Stelle in der Webapplikation, um einen Angriff erfolgreich durchführen zu können. Sie dagegen müssen jede dieser Schwachstellen im Voraus identifizieren und vollständig absichern.

Gleichzeitig werden besonders Webanwendungen gewöhnlich nicht isoliert betrieben, sondern sind mit zahlreichen anderen Systemen verbunden – darunter andere Applikationen, Datenbanken oder Benutzerverzeichnisse. Hat ein Angreifer erst einmal vollen Zugriff auf die Webanwendung, kommt er oft noch an weitere Systeme heran. Womöglich erhält er dadurch sogar Zugriff auf eigentlich geschützte Intranet-Anwendungen. Als (Java-)Entwickler müssen Sie verhindern, dass Ihre Webanwendung zum schwächsten Glied in der Kette wird und einem Angreifer Zugriff auf deren Daten oder gar dahinterliegende Systeme ermöglicht.

## 2.3 Wer muss sicher entwickeln?

Muss nun jeder Entwickler sichere Software entwickeln, oder genügt ein kleines speziell ausgebildetes Entwickler-Team, das alle Sicherheitsprobleme mitunter auch erst nachträglich behebt? Für beide Varianten gibt es Verfechter. Ich gehöre ganz klar zur ersten Gruppe. Jeder Entwickler muss zumindest über Grundkenntnisse in der sicheren Softwareentwicklung verfügen und diesen Prinzipien Tag für Tag folgen. Meiner Meinung nach ist das vergleichbar mit Grundkenntnissen in performance-orientierter Programmierung und einem guten Programmierstil.

Andernfalls benötigen Sie ein weiteres Team von Entwicklern, das im Nachgang Ihre fertig entwickelte Software überprüft und enthaltene Fehler korrigiert. Bei vergleichsweise simplen Programmierfehlern wie SQL Injection oder Cross-Site Scripting werden wohl nur die wenigsten Entwickler diese Aufgabe über längere Zeit erledigen wollen. Wer möchte schon die immer gleichen Bugs auf die immer gleiche Art beheben? Warum sollte man also nicht gleich allen Entwicklern eine

sichere Entwicklungsweise beibringen? Ich glaube nicht, dass Entwickler kein Interesse an sicherer Software haben. Genauso, wie Entwickler möglichst fehlerfreie Software ausliefern wollen, möchten Entwickler möglichst sichere Software erstellen. Sie müssen nur wissen, wie das geht.

*Jeder Entwickler benötigt Grundkenntnisse.*

Natürlich ist die Ausbildung bei entsprechend vielen Entwicklern teuer. Eine derartige Weiterbildung lässt sie auch nicht von heute auf morgen plötzlich nur noch sichere Software entwickeln, die Altlasten in Form von unsicheren Webanwendungen sind zunächst weiterhin vorhanden. Allerdings greift die simple Rechnung, was die (späte) Behebung eines Sicherheitsproblems per Hotfix im Vergleich zur Ausbildung aller Entwickler kostet, meist deutlich zu kurz. Der Ruf der Webanwendung oder gar des Unternehmens lässt sich nicht so einfach berechnen und schon gar nicht so einfach wiederherstellen. Per SQL Injection gestohlene Daten bleiben gestohlen. Per Cross-Site Request Forgery ausgeführte Transaktionen bleiben ausgeführt. Per Cross-Site Scripting publizierte Nachrichten bleiben publiziert. Das Ziel bei der sicheren Softwareentwicklung muss es daher sein, diese Probleme gar nicht erst entstehen zu lassen bzw. sie so schnell wie möglich, d. h. noch vor der Auslieferung, zu erkennen und zu beheben.

*Experten für Spezialthemen*

Ganz ohne Experten, z. B. für eine sichere LDAP-Anbindung, geht es selbst nach einer umfassenden Sicherheitsausbildung aller Entwickler nicht. Für derartige Spezialthemen genügt es, wenn einige wenige interessierte Entwickler über das dafür notwendige Expertenwissen verfügen. Diese kümmern sich dann um derlei Anforderungen und unterstützen die normalen Entwickler bei der Integration in die Webanwendung.

**Die Sicht des Managements**

Fairerweise muss man an dieser Stelle zumindest ein Stück weit zwischen der Produkt- und der Individualentwicklung unterscheiden. Zwar nicht aus Benutzer-, aber aus Managementsicht. Die Kosten für einen Security-Patch, selbst für eine tausend- oder gar millionenfach installierte Software wie einen Reader oder ein Anti-Viren-Produkt, sind für den Hersteller überschaubar. Die Software aktualisiert sich automatisch oder wird vom Anwender selbst manuell aktualisiert. Diese Kosten kann der Hersteller relativ konkret vorab schon abschätzen. Produkthaftung? (Meist) Fehlanzeige. Eine Individualsoftware hat es da deutlich schwieriger. Normalerweise haftet deren Hersteller umfassender; und für den Auftraggeber ist ein Datenverlust oder ein anderer entstandener Schaden einfacher nachweis- und einklagbar.

Ob sichere Softwareentwicklung notwendig ist oder nicht, darf jedoch niemals von der Größe des entwickelnden Unternehmens abhängig gemacht werden – egal ob Sie nun mit fünf Entwicklern eine Webanwendung entwickeln oder mit fünfhundert. Die erwartete Benutzeranzahl darf ebenfalls nicht die Entscheidungsgrundlage dafür bilden, ob eine Webapplikation sicher sein muss oder nicht. Sie muss es einfach immer sein. Zum einen wissen Sie vielleicht nicht, wie viele Benutzer Ihre Webapplikation letzten Endes verwenden werden, auch wenn es Schätzungen dafür gibt. Zum anderen können bereits 1000 gestohlene Kundendatensätze sehr wertvoll sein und die Schadensersatzforderungen das Unternehmen in die Insolvenz treiben.

## 2.4 Sicherheit in allen Phasen

Mit der Aussage, sichere Software benötige mehr als nur eine sichere Entwicklung, haben die Autoren des von Microsoft maßgeblich geprägten und entwickelten *Security Development Lifecycle* (SDLC, manchmal auch SDL)[3] zweifelsfrei recht. Es genügt aus diesem Grund nicht, nur die Entwickler für dieses Thema zu sensibilisieren und sie entsprechend auszubilden. Projektleiter, Anforderungsanalysten, Architekten, Tester, der Betrieb und weitere sind ebenso an der Entwicklung von sicheren Webanwendungen beteiligt wie die Entwickler selbst. Diese »Nichtentwickler« kümmern sich dabei um Themen wie zusätzliche Zeit und zusätzliches Budget für die Umsetzung, sie erstellen entsprechende Sicherheitsvorgaben für die Entwicklung oder entscheiden, welches Security-Framework verwendet werden soll.

Selbst wenn Sie nur die eigentliche Entwicklung von Webanwendungen mit den Augen eines Entwicklers betrachten, werden sicherheitsrelevante Entscheidungen in allen der drei Phasen *Design*, *Entwicklung* und *Konfiguration* getroffen (Abbildung 2-1).

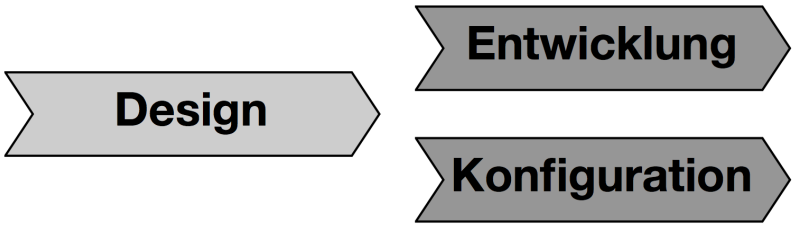


***Abb. 2-1***
*Sicherheit in allen Phasen*

[3] http://www.microsoft.com/security/sdl

*Designphase*

In der *Designphase* spezifizieren Sie zusammen mit dem Architekten die Webanwendung und legen beispielsweise fest, welche Rollen und Rechte vorhanden sind. Vor allem grundsätzliche Architekturentscheidungen werden hier getroffen. In dieser Phase gefällte Fehlentscheidungen sind später nur noch sehr schwer korrigier- und änderbar. Schlimmer noch: Selbst eine perfekt nach den Vorgaben entwickelte Webanwendung kann niemals ausreichend sicher sein, wenn in der Designphase Fehler, beispielsweise durch eine unzureichende Beachtung des Rollenkonzepts, begangen werden.

*Entwicklungsphase*

In der *Entwicklungsphase* geht es, wie der Name schon andeutet, um die Entwicklung der Webapplikation und der zugehörigen Tests (z. B. Unit- und Integrationstests). In dieser Phase programmieren Sie nach den festgelegten Vorgaben der Designphase. Und gerade hier passieren viele vermeidbare, mitunter auch sicherheitskritische Programmierfehler.

*Konfigurationsphase*

Die *Konfigurationsphase* lässt sich heutzutage eigentlich nicht mehr so richtig von der Entwicklungsphase trennen. Schließlich entwickeln und konfigurieren Sie meist parallel (abwechselnd) und liefern die Webanwendung in Inkrementen aus. Gleichzeitig werden viele Konfigurationsdateien mehr und mehr durch Annotationen ersetzt oder zumindest ergänzt. Zwei Trends sind hierzu feststellbar:

- Zum einen sind immer mehr Frameworks von vornherein sicher konfiguriert. Das ist definitiv ein Schritt in die richtige Richtung, der aber leider noch längst nicht überall allumfassend und vor allem konsistent ist. Im Idealfall sollte ohne eigene Konfiguration immer die sicherste Einstellung aktiv sein.
- Zum anderen aber verlagern sich gleichzeitig viele sicherheitskritische Programmierfehler in Richtung sicherheitskritische Konfigurationsfehler. Die Entwickler programmieren einfach weniger und konfigurieren mehr. Die absolute Anzahl an Fehlern wird also nicht zwangsläufig weniger, sondern verlagert sich in andere Bereiche.

Auch wenn alle drei Phasen gleichermaßen wichtig sind, ist dies ein Buch für Java-Entwickler. Der Fokus liegt daher ganz klar auf der Entwicklungsphase. Die Konfigurationsphase spielt an einigen Stellen ebenfalls eine größere Rolle; getrennt betrachten lassen sich diese Phasen wie gesagt ohnehin nicht mehr. Beginnen Sie mit diesen beiden Phasen, und weiten Sie die sichere Softwareentwicklung nach und nach auf alle Phasen des Projekts aus. Nur so haben Sie eine realistische Chance, eine sichere Webanwendung zu entwickeln.

In agilen Projekten sind die drei Phasen Design, Entwicklung und Konfiguration ebenso vorhanden, in Scrum beispielsweise innerhalb ei-