Java-Web-Security

Sichere Webanwendungen mit Java entwickeln

von
Dominik Schadow

1. Auflage

Java-Web-Security – Schadow

schnell und portofrei erhältlich bei beck-shop.de DIE FACHBUCHHANDLUNG

Thematische Gliederung:

Webprogrammierung

dpunkt.verlag 2014

Verlag C.H. Beck im Internet:
www.beck.de
ISBN 978 3 86490 146 1

Inhaltsverzeichnis: Java-Web-Security – Schadow

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).

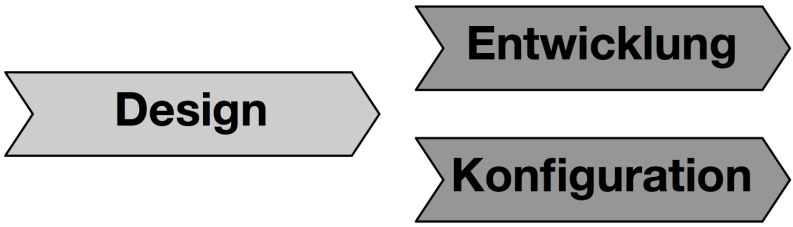

Abb. 2-1
Sicherheit in allen Phasen

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