SYRACOM

THE BUSINESS AND IT ARCHITECTS

A(GILE) SECURE DEVELOPMENT LIFECYCLE

Schneller zu sicherer Software

 Part of Consileon Group

Einleitung

Viele Unternehmen prüfen die Sicherheit ihrer Produkte erst am Ende der Entwicklung und unter hohem Zeitdruck. Dies geht zu Lasten der Gründlichkeit und der Produktqualität. Es bleibt kaum Zeit, die von den Sicherheitsexperten beanstandeten Mängel zu beheben sowie etwaige Änderungen des Designs oder der Konstruktion nachzuprüfen. Der resultierende Projektverzug und die damit einhergehenden finanziellen Verluste lassen sich vermeiden, wenn man die Sicherheit von Anfang an bedenkt.

Vorreiter eines solchen Vorgehens sind IT-Anbieter wie Microsoft, Google, Apple und Adobe. Bei ihnen fließen Sicherheitsaspekte in jede Phase der Softwareentwicklung ein. Microsoft hat dies exemplarisch dokumentiert [1]. Auch Führungskräfte anderer Unternehmen äußern sich zu diesem Thema, lassen sich dazu sogar interviewen (Beispiel: Adobe im Podcast „Risky Business", 2013).

Insbesondere die Software von Adobe galt in den letzten Jahren als Einfallstor für Schadprogramme. Sicherheitsexperten rieten, Adobe-Produkte zu entfernen oder zumindest zu deaktivieren. Immer wieder wurde auf Adobe ausgelegte Malware veröffentlicht, darunter ein Programm, das Laptop-Nutzer mit der eingebauten Kamera aufnahm und die Bilder ins Internet übertrug. Diese Verletzung der Privatsphäre zog enorme Kosten nach sich – infolge des Missbrauchs der Schwachstellen, deren Behebung, vor allem aber wegen des Imageschadens. Das Vertrauen in die Marke war erschüttert. Das Unternehmen stellte daraufhin seinen Entwicklungsprozess um. Bei jedem Schritt werden nun die einschlägigen Sicherheitsaspekte geprüft.

Viele Projektteams kennen das: Sicherheitsprüfungen am Ende der Entwicklung führen zu Nacharbeiten, die den Abschluss verzögern. Manchmal steht wegen der Konzernsicherheit gar die Systemfreischaltung auf dem Spiel. Doch damit nicht genug. Die in vielen Unternehmen anstehende Umstellung auf agile Entwicklungsmethoden wirft neue Sicherheitsfragen auf, denen sich Projektleiter und Entwickler stellen müssen. Wie lassen sich beide Probleme am besten gleichzeitig meistern?

Am Vorgehensmodell „Scrum" zeigen wir Möglichkeiten auf, Sicherheitsaspekte bei allen Schritten der Softwareentwicklung einzubeziehen. So entsteht ein sicherer, reproduzierbarer Entwicklungsprozess (Secure Development Lifecycle, SDL). Dieser Ansatz passt sowohl zu agilen wie zu klassischen Entwicklungsmethoden, auch wenn sich im Detail Abweichungen ergeben können.

Anhand einer Entwicklungsschleife, bei Scrum als Sprint bezeichnet, erläutern wir, an welchen Stellen man mit kleinen Veränderungen zeitig auf Sicherheitsaspekte eingehen kann. Dies kommt zum einen der Produktqualität zugute. Zum anderen wird Projektverzug durch Nacharbeit vermieden. Die potenziellen Kosten durch erfolgreiche Angriffe sinken deutlich.

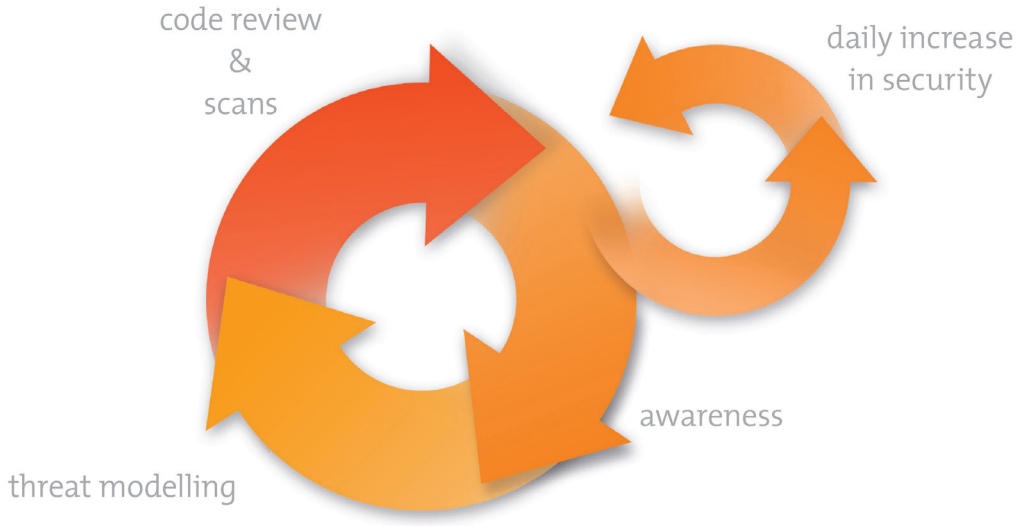

Abbildung 1:
Maßnahmen in einem Secure Development Lifecycle (SDL)

Scrum ist eine iterative Entwicklungsmethode (Abbildung 1). Und so sollte das Projektteam auch die Steigerung der Sicherheit des Softwareprodukts angehen. Der vorliegende Beitrag beschreibt Maßnahmen, mit denen Entwickler im Sprint die Sicherheit und die Qualität des Produkts stetig steigern. Neben dem Produkt selbst ist dabei die Besetzung des Projektteams von zentraler Bedeutung.

- Aus dem **Produkt** leiten sich die nötigen Sicherheitsmaßnahmen ab. Ein Übermaß ist hier ebenso kontraproduktiv wie mangelnde Sorgfalt.
- Im **Team** sieht das Vorgehensmodell Scrum drei Rollen vor: Entwickler, Product-Owner (Produkteigner, kurz PO) sowie den „Scrum-Master" (Prozessverantwortlicher, Mediator).

Team Die Mitglieder des Scrum-Teams sind in Sicherheitsthemen meist unerfahren. Traditionell versucht man diesem Manko mit Schulung und Zertifizierung abzuhelfen.

Nachhaltiger, aber auch schwieriger ist, von Anfang an Sicherheitsexperten ins Team einzubinden. Sie bereichern das Team nicht nur mit ihrem Wissen, sondern fungieren auch als allzeit wachsames Auge. Durch Arbeitstechniken wie die Paarprogrammierung, bei der jeweils zwei Entwickler in wechselnder Paarung am selben Rechner arbeiten, lässt sich die Kenntnis der Schwachstellen und ihrer Vermeidung sukzessiv an alle Teamkollegen weitergeben.

Je nach Schutzbedarf des Produkts kann der Einsatz der Sicherheitsexperten befristet werden. Alternativ können externe Kollegen zeitweilig unterstützen. Bei Scrum lässt sich ein solches Vorgehen in Sprints beispielsweise zu Penetrationstests oder Codereviews in dedizierten Zeitfenstern, „Spikes" genannt, abbilden. Zudem sollte der PO erwägen, die Entwicklung einzelner Komponenten komplett an die Sicherheitsexperten zu delegieren.

Sprintplanung

Sprintplanung I:

Die Planung eines Sprints gliedert sich in zwei Phasen. Zunächst stellt der PO dem Team den Gegenstand des Sprints aus Nutzersicht vor. Neben der gewünschten Funktion nennt diese „User-Story" die betreffenden Abnahmekriterien. Hier kann der PO vorgeben, dass das Produkt von typischen Schwachstellen frei sein muss. Das Open Web Application Security Project (OWASP) und das SANS-Institut publizieren Listen akuter Sicherheitslücken [2, 3]. Zur Unterstützung der Entwickler publiziert OWASP unter anderem Spickzettel („Cheat Sheets") zu den jeweiligen Sicherheitslücken. In den gängigsten Programmiersprachen skizzieren diese Hinweise sichere Lösungen zu kritischen Schwächen [4].

Auf der Checkliste zur Abnahme der Software („Definition of Done" in der Scrum-Nomenklatur) könnte demnach stehen, dass der Programmcode erst als fertig gilt, wenn alle Schwachpunkte anhand der OWASP-Spickzettel beseitigt sind.

Darüber hinaus liegt es nahe, in der ersten Phase der Sprintplanung die Risiken jeder User-Story zu beschreiben (Threat-Modelling). Dies obliegt nicht allein dem PO, sondern sollte vom Team als ganzheitliche Aufgabe wahrgenommen werden. Vorgehensweisen wie DREAD [5] oder STRIDE [6] helfen bei der Risikobewertung. Eine solche Analyse liefert auch Hinweise zur Implementierung. Diese werden in der zweiten Planungsphase als separate Arbeitsschritte (Tasks) formuliert.

Über mehrere Sprints hinweg zeigen die Analysen zudem, wie sich das Gesamtrisiko des Produkts darstellt. So kann der PO sowohl prozessuale wie technische Angriffsflächen erkennen, bevor diese von Hackern ausgenutzt werden.

Ergänzend fassen Großunternehmen, in denen viele Projekte in einer komplexen Systemlandschaft parallel laufen, Systeme mit gleichem Schutzbedarf zu Risikogruppen zusammen. Ein Webshop etwa ist anderen Risiken ausgesetzt als der interne Druckserver. Durch die Gruppierung nach Risiko lassen sich Sicherheitskriterien sowohl vereinheitlichen als auch übertragen. Auch kann das Unternehmen auf diese Weise die maximal zulässige Exposition vorgeben.

Alternativ formuliert der PO zur Sicherheitsanalyse eine eigene User-Story und leitet daraus gemeinsam mit dem Team Korrekturmaßnahmen ab. Es empfiehlt sich, dazu einen festen Zeitrahmen („Timebox“ oder „Spike“) vorzugeben.

Da die Komponenten der Software immer wieder auf dieselben Schwachstellen zu testen sind, sinkt bei allen Beteiligten im Zuge der Iteration tendenziell die Aufmerksamkeit. Durch Rotation im Team und gelegentliches Einbinden externer Prüfer kann man diesem Ermüdungseffekt entgegensteuern.

Sprintplanung II:
In der zweiten Phase der Sprintplanung konkretisiert das Team die User-Storys zu einzelnen Tasks. Dabei legt es auch Details der Implementierung fest. Sicherheitsaspekte können hier ebenfalls als Tasks formuliert werden, zum Beispiel als Codereview. Um zu vermeiden, dass Sicherheitslücken aus Betriebsblindheit übersehen werden, sollten solche Reviews anderen Entwicklern obliegen als denen, die an der jeweiligen Story gearbeitet haben.

Ein Task, der keinen direkten Bezug zu einer User-Story haben muss, ist das Untersuchen der Applikation mit Softwarescannern wie Nessus, ZAP, BURP oder Metasploit. Diese Scans setzt man am besten unmittelbar vor dem finalen Sprintreview an, wenn alle neuen Funktionen implementiert sind. Eine weitere Möglichkeit besteht darin, zu einer User-Story einen eigenen Task anzulegen. Dieser sieht dann eine Prüfung der einzelnen Softwaremodule vor.

Die Kooperation mit der Sicherheitsabteilung beziehungsweise dem zuständigen Teamkollegen sollte sich wie folgt gestalten: Während der Entwickler einen Modultest entwirft, der eine Liste mit Aufrufen und dem jeweils erwarteten Ergebnis abarbeitet, listet der Sicherheitsexperte denkbare Angriffe und die zu erwartende Reaktion auf. Im Idealfall legt das Testsystem bei kritischen Treffern im Issuetracker des Teams automatisch eine Meldung (Ticket) an, aus der hervorgeht, welche neue Lücke gefunden wurde.

Entwicklung im Sprint

Viele Werkzeuge zur Analyse der Sicherheit lassen sich in Continuous-Integration- (CI) oder Continuous-Deployment-Systeme (CD) integrieren. Dies geschieht in Form separater Tests oder eigener Buildschritte. Das Ergebnis des Scans wird automatisch ausgewertet, das Team somit frühzeitig auf etwaige Schwachstellen aufmerksam. Das Buildresultat wird auf „verfehlt" (failed) gesetzt, sobald eine Lücke gefunden wurde. So lässt sich frisch gebaute und installierte Software im CI- oder CD-Zyklus mit geringem Aufwand einem vereinfachten Penetrationstest unterziehen.

Zwei Ansätze kommen dazu in Frage:

Erstens kann man den Quellcode nach Ablage im Repository mit Werkzeugen wie Fortify oder Checkmarx prüfen. Daneben besteht die Möglichkeit, den Code in der Entwicklungsumgebung „IntelliJ IDEA" von JetBrains schon vor der Archivierung auf riskante Programmierfehler zu untersuchen und so die Aufnahme von Code mit Schwachstellen in das Repository zu verhindern. Anfangs mag dies das Arbeitstempo der Entwicklungs- und Testumgebungen bremsen. Dieser Effekt verflüchtigt sich jedoch. Die Entwickler lernen schnell, einmal gefundene Fehler zu vermeiden.

Der zweite Ansatz besteht darin, die Software nach der Installation zur Laufzeit zu scannen. Dazu werden Tools wie ZAP, BURP, Nessus oder Metasploit als Schritte nach dem Buildprozess in den CI- oder CD-Zyklus integriert. Auch

hier sollte das Aufspüren einer Schwachstelle den Abschluss des Buildprozesses blockieren.

Die genannten Tests prüfen die Sicherheit des Softwarestandes nur partiell. Dennoch bedeuten sie gegenüber der gängigen Projektpraxis einen großen Fortschritt. Ergänzend lässt sich das Ergebnis automatisch in ein Ticketsystem laden. Die Entwickler stellen dadurch mit minimalem Zeitverzug fest, ob die jüngsten Änderungen neue Sicherheitslücken aufreißen. Durch die automatische Erfassung als Issue ist jederzeit belegt, welche Schwachstellen bekannt und noch zu beheben sind. Dies wiederum erweitert die Planungsgrundlage des PO. Abbildung 2 stellt die besprochenen Schritte im Ablauf dar.

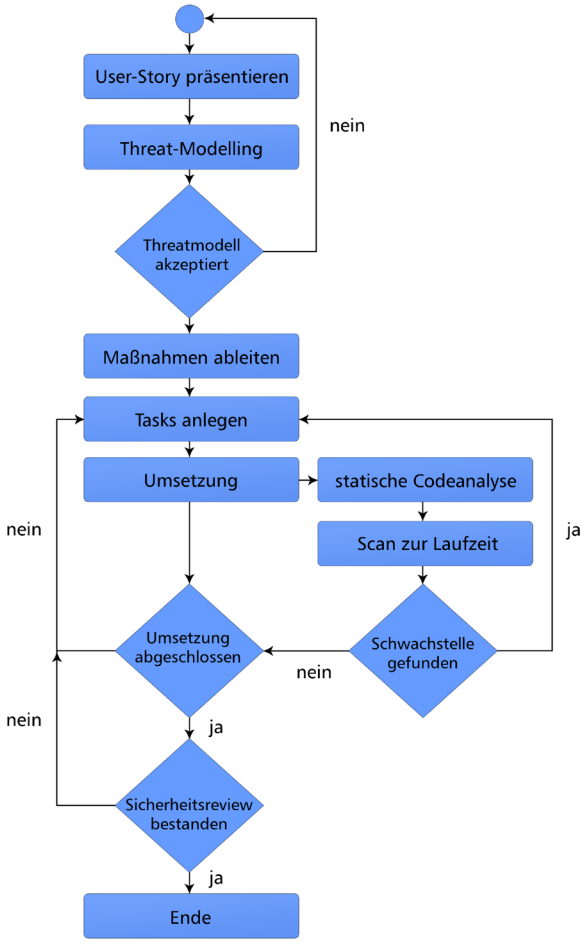

Abbildung 2:
Schritte eines SDL- Durchgangs

Ungeachtet des zur Prüfung des Quellcodes verwendeten Werkzeugs empfiehlt es sich, von Zeit zu Zeit einen Sprint in ein komplettes Software-Assessment zu investieren. Bewertungen dieser Art sind beispielsweise in OpenSAMM [7] oder in BSIMM [8] dargelegt. Neben der formalen Qualität der Software werden dabei die Erfüllung der funktionalen Anforderungen und

Sprintreview

die Gesamtarchitektur beurteilt. So entsteht ein umfassendes Bild der Reife und Sicherheit des Produkts.

Nach jedem Sprint präsentieren Team und PO den übrigen Stakeholdern das Ergebnis. Den Aufwand zur Härtung der Software zu begründen und den Härtegrad zu zeigen, gestaltet sich meist schwierig. Alle hier beschriebenen Schritte leisten dazu jedoch einen gut visualisierbaren Beitrag. Als überzeugend erweist sich oft eine Demonstration des Schutzniveaus vor und nach dem Sprint.

Virtuelle Patches und Releases

Unter einem virtuellen Patch versteht man die Installation, Konfiguration und den Betrieb von Software, die eine gerade erkannte Sicherheitslücke abschirmt, Einbruchsversuche registriert und abwehrt. Durch Kombination einer Web-Application-Firewall (WAF), also einer Firewall nicht auf Netz-, sondern auf der Anwendungsebene, mit Intrusion-Detection- (IDS) und Intrusion-Prevention-Systemen (IPS) lassen sich Schwachstellen relativ schnell sichern, bzw. Einbruchsversuche erkennen und blockieren. Allerdings sind solche Installationen als Dauerlösung ungeeignet, da sie ständiger Wartung bedürfen.

Mit einer iterativen Entwicklungsmethodik wie Scrum kann der PO kurzfristig auf ungeplante Ereignisse reagieren – etwa wenn sich im produktiven Einsatz der neuen Software weitere Schwachstellen offenbaren. Der erste Schritt dazu ist, eine WAF aufzusetzen und so zu konfigurieren, dass sie den unbefugten Zugriff über die jeweilige Lücke vereitelt. Parallel sollte das Team mit der Entwicklung des Korrekturcodes beginnen.

Das bringt uns zur Releaseplanung. Zur Beseitigung nachträglich erkannter Schwachstellen einer bereits produktiv eingesetzten Software ist es ratsam, einen festen Zeitpuffer vorzusehen.

Bei konsequentem Threat-Modelling kann man aus dem Katalog noch ausstehender Funktionen („Product-Backlog“ im Scrum-Jargon) ablesen, welche Schritte zur Konfiguration und Härtung der Betriebsumgebung inklusive WAF, IDS und IPS sinnvoll sind. Da das Scrum-Team zumindest an der WAF-Konfiguration mitwirken sollte, hängt die Releaseplanung auch von seiner Einbindung in den Betrieb ab.
Ferner sollte die Versionsplanung das unter „Entwicklung im Sprint“ ange-

sprochene Assessment nach OpenSAMM oder BSIMM berücksichtigen. Dabei wird das Produkt vom Code über die Architektur bis zur Erfüllung der Anforderungen durchgeprüft. Aus dieser ganzheitlichen Analyse ergeben sich oft weitere User-Storys und Aufgaben für die Entwickler, die es einzuplanen gilt.

Fazit Bei der agilen Entwicklung wird der Funktionsumfang des Produkts in kleinen, leicht prüfbaren Schritten ausgebaut. Durch Analyse des Quelltextes und durch Sicherheitstests zur Laufzeit lassen sich Schwachstellen aufspüren und beseitigen. Dabei helfen in den Entwicklungszyklus integrierte, automatisierte Tools.

Werkzeuge sind jedoch nur so gut wie die Menschen, die sich ihrer bedienen. Je nach Schutzbedarf ist daher zu entscheiden, ob und wie viele Sicherheitsexperten zum Team gehören oder dieses temporär unterstützen sollen. Die Prüfung der Sicherheit darf sich indes nicht auf den Code beschränken. Prozessmängel, unsichere Funktionen und weitere Risiken ermittelt man am einfachsten durch Threat-Modelling. Dazu kann jedes Teammitglied beitragen.

Das hier umrissene Vorgehen eignet sich nicht nur für Scrum-Projekte. Vom Threat-Modelling im Zuge der Formulierung der Anforderungen über das Lösungskonzept bis zum Rollout passen die Maßnahmen auch zum Wasserfallmodell. Dazu müssen unter anderem die regelmäßigen Scans des Continuous Deployment als separate Testphasen realisiert werden. Analysen des Codes vor oder nach der Ablage im Repository lassen sich ohne weiteres in Wasserfall-Projekte einbauen. Sie bilden im SDL eine wichtige Etappe.

Die ständige Kontrolle der Ergebnisse der Sicherheitsanalyse und daraus abgeleiteter Maßnahmen ist ein Eckpfeiler des SDL. Marktübliche Tools können mit geringem Aufwand integriert werden. Dies gilt sowohl bei klassischen wie bei agilen Vorgehensmodellen.

Mit einer sicheren Entwicklung allein ist es allerdings nicht getan. Noch bevor die Software die Betriebsumgebung erreicht, lohnt es sich, über flankierende Schutzmaßnahmen wie virtuelle Patches nachzudenken. Gerade entdeckte Sicherheitslücken können mit WAFs und anderen Systemen abgedichtet werden, bis die Entwickler mit dem nächsten Sprintergebnis den Korrekturcode liefern.

Links und Literatur

[1] http://www.microsoft.com/security/sdl/default.aspx
[2] https://www.owasp.org/index.php/Top_10_2013
[3] http://www.sans.org/critical-security-controls
[4] https://www.owasp.org/index.php/Cheat_Sheets
[5] http://blogs.msdn.com/b/david_leblanc/archive/2007/08/13/dreadful.aspx
[6] http://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx
[7] http://www.opensamm.org
[8] http://bsimm.com

SYRACOM

THE BUSINESS AND IT ARCHITECTS

Ansprechpartner

Dr. Markus Maria Miedaner
Senior Consultant
Certified Information System Security
Professional (CISSP)
markus.miedaner@syracom.de

Frank Polscheit
Managing Consultant
frank.polscheit@syracom.de

Otto-von-Guericke-Ring 15
65205 Wiesbaden
Tel: +49 6122 9176-0
www.syracom.de