Unternehmen, die Sicherheitslücken in Anwendungen vor deren Auslieferung oder Implementierung schließen, profitieren von enormen Kosteneinsparungen – intern und für Kunden, Geschäftspartner und andere Stakeholder, die mit ihrer Software arbeiten.

Dezember 2009

Rational. software

Sicherer Quellcode

Integration von Schwachstellentests im Quellcode in den Lebenszyklus bei der Softwareentwicklung

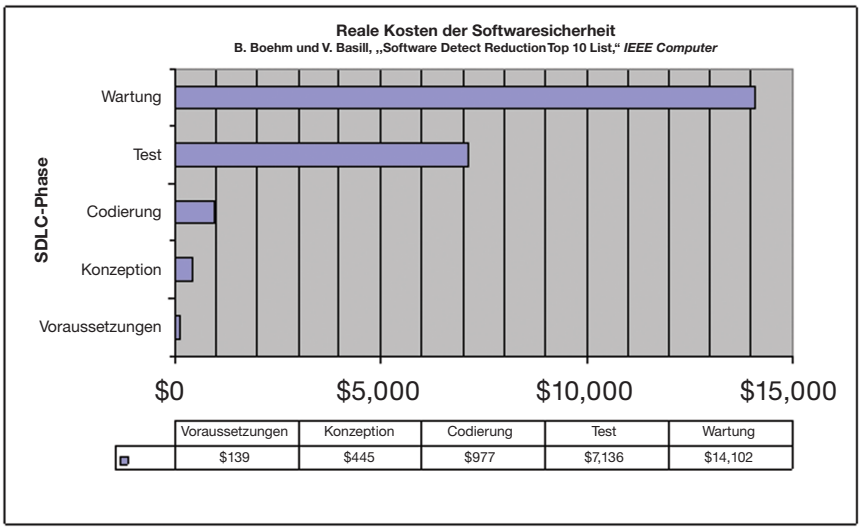
Ryan Berg
IBM Senior Security Architect
IBM Software Group

Unzählige Studien und Empfehlungen von Analysten weisen auf die Bedeutung hin, die Sicherheit bereits während der Softwareentwicklung zu verbessern, anstatt Sicherheitslücken in der Software zu schließen, die erst nach umfassender Verwendung und Implementierung festgestellt werden. Die Gründe hierfür liegen auf der Hand.

Für die Softwareanbieter ergeben sich durch Sicherheitslücken, die in ihren Produkten gefunden werden, sowohl direkt als auch indirekt Kosten. Die erneute Bereitstellung von Mitarbeitern aus der Entwicklungsabteilung für die Erstellung und Verteilung von Patches kann Softwareanbieter in vielen Fällen Millionen kosten. Die Ausnutzung einer einzigen Sicherheitslücke hat in Unternehmen auf der ganzen Welt in einigen Fällen bereits zu Umsatzverlusten in Milliardenhöhe geführt. Anbieter, denen die Schuld an Sicherheitslücken im Quellcode ihres Produkts gegeben wird, müssen den Verlust von Glaubwürdigkeit, Imageschäden für ihr Markenprodukt und Wettbewerbsnachteile befürchten. Laut einer Studie von Carnegie Mellon aus dem Jahr 2005 sank der Aktienkurs von Anbietern um durchschnittlich 0,63 Prozent im Vergleich zum NASDAQ (NASD Automated Quotations), nachdem eine Sicherheitslücke in ihrer Software festgestellt wurde.[1]

Es gibt allerdings keine ähnlich detaillierten Studien über Sicherheitslücken in angepasster Unternehmenssoftware, die entweder intern oder im Rahmen einer Outsourcinglösung entwickelt wurde. Es herrscht große Einigkeit darüber, dass Sicherheitslücken mit umso geringerem Kostenaufwand behoben werden können, je frühzeitiger sie im Lebenszyklus festgestellt werden. Wie im folgenden Diagramm dargestellt, das von B. Boehm und V. Basali in IEEE Computer veröffentlicht wurde, haben Untersuchungen ergeben, dass das Beheben von Softwarefehlern nach der Implementierung 100 mal teurer ist im Vergleich zu den Kosten, die für die Behebung dieser Fehler in den ersten Entwicklungsphasen anfallen.[2] Bei Sicherheitsfehlern liegen die Kosten für deren Behebung zu einem späten Zeitpunkt häufig noch deutlich darüber, da hier nicht nur die Sicherheitslücken geschlossen werden müssen, sondern die Ausnutzung der Schwachstellen auch zu Datendiebstählen, Sabotage oder anderen Angriffen führen kann.

Die möglichst frühzeitige Ermittlung und Beseitigung von Sicherheitsproblemen kann dazu beitragen, die Entwicklungskosten zu reduzieren und die Softwarequalität zu verbessern. Es ist bekannt, dass dies die Implementierung und Verwendung von Sicherheitstools für den Quellcode während des gesamten Lebenszyklus in der Softwareentwicklung ermöglicht.



Automatisierte Quellcodeanalysen sind weithin als die effektivste Methode für Sicherheitstests zu einem frühen Zeitpunkt im Lebenszyklus anerkannt, da beliebige Teile des Codes analysiert werden können, ohne eine vollständige Anwendung zu benötigen. Die besten Lösungen auf der Grundlage dieser Technologien liefern die wertvollsten Ergebnisse, indem bei Schwachstellen auf die genaue Codezeile hingewiesen wird und Informationen über die Art der Schwachstelle, deren Bedeutung und die Vorgehensweise zur Fehlerbehebung mitgeliefert werden. Das ethische Hacken ist ebenfalls ein wichtiges Element der Softwaresicherheit, dessen Bedeutung allerdings erst zu einem späteren Zeitpunkt im Lebenszyklus zum Tragen kommt, wenn es für eine vollständige Anwendung über eine funktionale Schnittstelle eingesetzt werden kann.

Hindernisse auf dem Weg zur Sicherheit

Die größte Hürde, die Sicherheitstests im Lebenszyklus der Softwareentwicklung erschwert, ist üblicherweise die Diskrepanz zwischen Entwicklung und Sicherheitsniveau. Nur selten verfügen Einzelpersonen, Arbeitsgruppen oder ganze Unternehmen über Fachwissen aus beiden Bereichen. Daher ergeben sich nur sehr wenige Synergieeffekte. Während in der Entwicklung das Hauptaugenmerk auf der Funktionalität eines Produkts und der Bereitstellung zum vorgesehenen Zeitpunkt liegt, sind Sicherheitsanalysten häufig

erst nach der vollständigen Entwicklung und Implementierung von Anwendungen mit der Beseitigung von Sicherheitslücken und der Integration von Sicherheitsfunktionen beschäftigt. Damit während des Entwicklungsprozesses entstandene Schwachstellen effektiv beseitigt werden können, muss eine Zusammenarbeit zwischen den Teams aus den Bereichen Entwicklung und Sicherheit erreicht werden. In allen Fällen ist die Unterstützung durch die Unternehmensführung für eine größere Bedeutung von Sicherheitsfunktionen in der Entwicklung von entscheidender Bedeutung.

Neben betrieblichen Hindernissen kann auch ein allgemeines Zögern im Hinblick auf Änderungen oder Korrekturen an bestehenden Prozessen in der Softwareentwicklung zu Verzögerungen bei der Umsetzung von Sicherheitstests beitragen. Ein einfaches Verständnis der zu erreichenden Vorteile auf Unternehmensebene sind jedoch häufig Motivation genug, um Dinge voranzutreiben. Ähnlich diesem konzeptionellen Hindernis gibt es viele falsche Vorstellungen über die Integration von Sicherheitsanalysen während der Entwicklung, die korrigiert werden müssen, bevor eine Initiative in Bewegung kommt.

Annahme:
Der Zeitplan in der Entwicklung kann nicht weiter ausgedehnt werden, auch nicht, um Sicherheitsrisiken einzugrenzen.

Fakt:
Im Entwicklungszyklus kann es anfänglich zu Versäumnissen kommen, insbesondere, da sich jeder Einzelne mit dem neuen System vertraut machen muss. Es ist dennoch die Methode mit dem geringsten Zeitaufwand zur Vermeidung von Softwarerisiken. Der Prozess trägt letztendlich zu einem kürzeren Entwicklungszeitraum bei, da die Entwickler bewährte und sichere Codierungsverfahren kennenlernen. Die einzige schnellere Alternative besteht darin, überhaupt nichts zur Verbesserung der Softwaresicherheit zu unternehmen. Dies können sich die meisten Unternehmen langfristig sicherlich nicht leisten.

Annahme:
Wir führen bereits eine Prüfung durch Kollegen durch (Peer Review), zusätzliche Prüfungen des Sicherheitscodes sind daher nicht erforderlich.

Fakt:
Eine Prüfung durch Kollegen ist kein gleichwertiger Ersatz für eine Sicherheitsprüfung. Bei dieser Art der Prüfung wird die Software normalerweise nur auf funktionale Fehler hin überprüft. Viele der maßgeblicheren Sicherheitslücken und Schwachstellen im Design werden hierbei übersehen, es sei denn, die Prüfer sind umfassend mit dem Bereich Anwendungssicherheit vertraut. In vielen Fällen können gutgemeinte und ohne funktionale Fehler implementierte Benutzeranforderungen zu den größten Sicherheitsrisiken führen.

Zentrale Verantwortlichkeiten

Für viele Unternehmen stellt die Ermittlung der am besten geeigneten Methode und Ressourcen zur Implementierung von Quellcodeanalysen in den Lebenszyklus der Softwareentwicklung weiterhin eine große Herausforderung dar. Bei den drei Modellen, die in diesem White Paper vorgestellt werden, handelt es sich um gängige Szenarios, die derzeit eingesetzt werden, um Sicherheitslücken in der Softwareentwicklung zu beseitigen. Mit diesen Modellen können Kriterien für die Beurteilung von Zielen, Ressourcen, Hindernissen und letztendlich das am besten geeignete Konzepte für einzelne Unternehmen festgelegt werden.

In diesem White Paper wird kein neuer Prozess zur Darstellung von Risiken erläutert. Vielmehr werden eine Reihe von Workflowmodellen dokumentiert, um eine Anleitung zu geben, wie automatisierte Quellcodeprüfungen in einen bestehenden Entwicklungsprozess integriert werden können.

Obwohl sich Entwicklungsunternehmen und -prozesse natürlich durch jeweils eigene eindeutige Merkmale auszeichnen, werden anhand dieser Modelle einheitliche Elemente vorgestellt, die eingesetzt werden müssen, um effektive Sicherheitstests zu erreichen. Bei den nachfolgenden Erläuterungen müssen durch die vorhandenen Mitarbeiter oder Experten, die während der Implementierung hinzugezogen werden, folgende primäre Funktionen beachtet werden:

Festlegung von Sicherheitsanforderungen: Ein Manager oder eine zentrale Stelle mit sicherheitsspezifischem Know-how definiert die Schwachstellen und legt fest, wie deren Dringlichkeit auf der Grundlage der Geschäftsanforderungen zu beurteilen ist.

Konfiguration der Analyse: Für die Anpassung des Tools zur Quellcodeanalyse an bestehende Richtlinien werden interne Definitionen verwendet.

Prüfung des Quellcodes: Das Tool zur Quellcodeanalyse wird für die Zielanwendung oder Teile der Anwendung ausgeführt, um Schwachstellen aufzudecken.

Priorisierung von Ergebnissen: Mitarbeiter mit sicherheits- und anwendungsspezifischem Fachwissen priorisieren den Ablauf zur Fehlerbehebung auf der Grundlage der Ergebnisse, damit zunächst Schwachstellen mit einem hohen Sicherheitsrisiko beseitigt werden.

Beseitigung von Schwachstellen: Schwachstellen werden durch das Umschreiben von Code, das Entfernen fehlerhafter Komponenten und das Hinzufügen sicherheitsspezifischer Funktionen beseitigt.

Überprüfung von Korrekturen: Der Code wird erneut geprüft und untersucht, um sicherzustellen, dass durch die Änderungen am Code alle Schwachstellen ohne Beeinträchtigung der Funktionalität der Anwendung beseitigt wurden.

I: Unabhängiges Modell

Dieses Modell wird bei der Einführung des Konzepts der Quellcodeanalyse am häufigsten verwendet. Bei diesem Modell ist jeder Entwickler dafür verantwortlich, Code auf Sicherheitslücken hin zu analysieren, die am dringendsten erforderlichen Korrekturen zu ermitteln und bereitzustellen und sicherzustellen, dass alle Schwachstellen beseitigt wurden. Je nach Unternehmen und Anwendungsanforderungen werden diese Sicherheitsmaßnahmen normalerweise in regelmäßigen Abständen während des Entwicklungszyklus durchgeführt. Softwareentwickler prüfen den eigenen Code, nachdem sie Änderungen vorgenommen oder Zeilen hinzugefügt haben, und beseitigen alle Schwachstellen, bevor sie die Dateien wieder in das Kontrollsystem für den Quellcode einchecken. Idealerweise prüft jeder Einzelne die gesamte Anwendung, um den jeweiligen Code im Gesamtzusammenhang zu analysieren, und um den Datenfluss und mögliche Sicherheitslücken für alle komplexen Interaktionen der unterschiedlichen Dateien und Funktionen nachzuvollziehen. Dieser Vorgang kann extrem viel Zeit in Anspruch nehmen, es sei denn, die gesamte Codebasis ist relativ klein.

Wie funktioniert es?

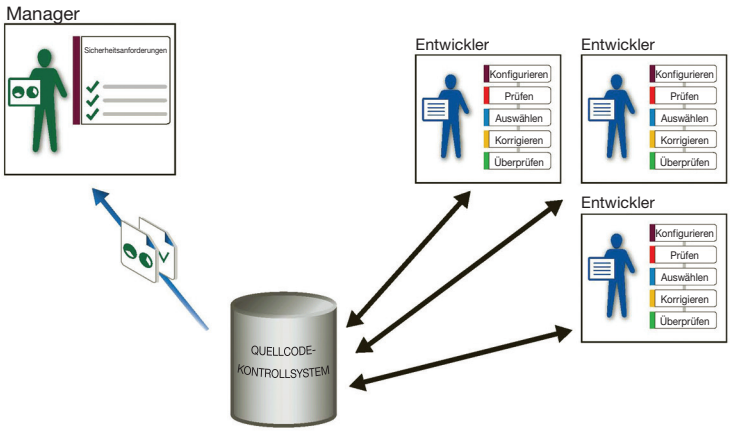
Dieses Modell sollte während der Entwicklungsphase im Rahmen eines vorhandenen Lebenszyklus bei der Softwareentwicklung eingeführt werden. Hierfür müssen Entwickler hinzugezogen werden, um nicht zur bei der Identifizierung von Sicherheitslücken und der Festlegung einer geeigneten Priorisierung, sondern auch bei der Bestimmung, wie diese bestmöglich behoben werden, auf sicherheitsspezifisches Know-how zurückgreifen zu können. Für die Bereitstellung grundlegender Sicherheitsanforderungen und Anleitungen zur Entscheidungsfindung sind normalerweise einige Vorgaben durch das Management erforderlich (z. B. im Hinblick auf die Beurteilung der Dringlichkeit von Schwachstellen bei Webschnittstellen im Gegensatz zu Schwachstellen in einer Datenbankanwendung). Wenn Sicherheitsrichtlinien nicht zentral festgelegt werden, müssen die Entwickler selbst entscheiden, welche Faktoren eine Sicherheitslücke charakterisieren oder eine Korrektur rechtfertigen.

Das Management kann den Fortschritt bei der Entwicklung nachvollziehen und Verzögerungen erkennen, die durch diese Sicherheitsmaßnahmen verursacht werden. Im Allgemeinen sind bei diesem Modell keine allgemeinen Berichte zu gefundenen und behobenen Schwachstellen vorgesehen, da diese Schwachstellen theoretisch vor der Eingabe in eine zentrale Berichtsstruktur behoben wurden.

Unternehmen, die Sicherheitslücken in Anwendungen vor deren Auslieferung oder Implementierung schließen, profitieren von enormen Kosteneinsparungen – intern und für Kunden, Geschäftspartner und andere Stakeholder, die mit ihrer Software arbeiten.

In welchen Fällen funktioniert es?

Unternehmen mit kleinen Entwicklungsteams und wenig komplexen Anwendungen haben den größten geschäftlichen Nutzen durch dieses unabhängige Modell. In diesen Fällen ist es erheblich einfacher, ausreichende Schulungsmaßnahmen und Anleitungen zur Verfügung zu stellen, damit die Sicherheitsmaßnahmen der Entwickler zu einer effektiven Fehlerbehebung führen. Obwohl die Nachverfolgung von Verbesserungen bei diesem Modell schwierig ist, kann man davon ausgehen, dass Entwickler Erfahrungen aus bisherigen Sicherheitsmaßnahmen sammeln und beim Schreiben neuen Code gängige Fehler vermeiden. Diese Vorteile und die relativ niedrigen Investitionen, die zur Umsetzung des unabhängigen Modells erforderlich sind, haben sich bei der Einführung von Sicherheitsfunktionen in kleinen Entwicklungsunternehmen als durchführbare Methode bewährt. Das Modell bietet sich auch für einfache Projekte von Systemintegratoren an, solange Analyse und Fehlerbehebung von nur wenigen Entwicklern durchgeführt werden.



Unternehmen, die Sicherheitslücken in Anwendungen vor deren Auslieferung oder Implementierung schließen, profitieren von enormen Kosteneinsparungen – intern und für Kunden, Geschäftspartner und andere Stakeholder, die mit ihrer Software arbeiten.
Seite 8

Workflow:

Manager

- Definition der Sicherheitsanforderungen

Entwickler

- Auschecken des Codes
- Hinzufügen oder Ändern von Funktionen
- Konfiguration der Scanner
- Prüfen der Anwendung
- Priorisierung der Ergebnisse
- Durchführung erforderlicher Korrekturen
- Ausführung des Scans zur Überprüfung der Korrekturen
- Einchecken aktueller Codeänderungen

Manager

- Prüfen der Berichte aus der Entwicklung

Für Entwicklungsunternehmen mit umfassendem Know-how im Sicherheitsbereich lässt sich eine Abwandlung dieses Modells durch Herausgreifen einzelner oder einer geringen Anzahl von Entwicklern mit Kenntnissen im Sicherheitsbereich oder Zugang zu Schulungsmaßnahmen erstellen. Dadurch übernehmen diese Entwickler beim Ermitteln und Korrigieren von Sicherheitslücken die Rolle eines Mentors und unterstützen ihre Kollegen dabei, ein besseres Verständnis für die Grundlagen sicherer Entwicklungsprozesse zu erlangen.

In welchen Fällen funktioniert es nicht?

Das unabhängige Modell lässt sich nicht auf kleine Anwendungen oder kleine Teams anwenden, die auf die Prüfung von Quellcode spezialisiert sind. Die meisten Unternehmen können sich nicht darauf verlassen, dass Entwickler über genügend Fachwissen verfügen, um wichtige sicherheitsrelevante Entscheidungen treffen zu können, oder ausreichend geschult sind, um diesen Kenntnisstand zu erreichen, für den häufig

ein unverhältnismäßig hoher Personalaufwand erforderlich ist. Selbst wenn Entwickler diesen Kenntnisstand erreichen, stellt sich dieser Ansatz aufgrund des Mangels an zentralen Kontrollfunktionen und Prozessen häufig als ineffizient heraus, der redundante Tätigkeiten der Entwickler nach sich zieht, die z. B. dieselben Codebasen gleichzeitig prüfen und möglicherweise an der Korrektur derselben Schwachstellen arbeiten. Dieses Modell eignet sich auch nicht aufgrund von Schwierigkeiten bei der Erstellung und Umsetzung sicherer Codierungsrichtlinien, die nicht nur für einige wenige Entwickler gelten. Ohne überprüfbare Standards, die im gesamten Unternehmen definiert und umgesetzt werden (z. B. welcher Grad an Verschlüsselung verwendet oder wie Eingaben geprüft werden sollen), setzen sich immer die von einzelnen Prüfern bevorzugten Methoden als Standardverfahren durch, die wiederum zu Uneinheitlichkeiten und in vielen Fällen zu einem niedrigen Sicherheitsniveau führen.

Ein weiterer Nachteil ist die fehlende Möglichkeit, aussagekräftige Projektdaten aus Unternehmenssicht nachvollziehen zu können, z. B. Anzahl und Art der ermittelten Schwachstellen, Verbesserungen bei der Anwendungssicherheit im Lauf der Zeit und Return on Investment für die verwendeten Sicherheitsressourcen. Wenn Entwickler die richtigen Kenntnisse und Tools zur Hand haben, ist es sehr wahrscheinlich, dass Schwachstellen schon während der Entwicklung beseitigt werden. Ohne Berichte und Artefakte, in denen diese Verbesserungen zum Ausdruck kommen, gibt es jedoch keine praktikable Möglichkeit, um Führungskräften, Auditoren, Kunden, Geschäftspartnern und anderen Beteiligten den geschäftlichen Nutzen zu verdeutlichen.

Bewährte Verfahren

Die bestmöglichen Ergebnisse lassen sich beim unabhängigen Modell durch folgende Maßnahmen erreichen:

Legen Sie eine Reihe quantifizierbarer und umsetzbarer Sicherheitsanforderungen als Anleitung bei Maßnahmen zur Fehlerbehebung fest. Führen Sie darüber hinaus Sicherheitsprüfungen bei Entwicklern durch, um zu prüfen, ob mit allen Korrekturen im Sicherheitsbereich Schwachstellen effektiv beseitigt werden, ohne dass sich dies nachteilig auf die Funktionalität auswirkt.

Benennen und schulen Sie einen Mitarbeiter mit sicherheitsspezifischen Kenntnissen aus der Entwicklungsabteilung, damit dieser als Mentor fungieren kann. Dieser Mitarbeiter leitet wiederum andere Entwickler bei Analysen, Priorisierungen und Sicherheitsprüfungen für Fehlerkorrekturen an.

II: Verteiltes Modell

Ähnlich wie beim unabhängigen Modell wird auch beim verteilten Modell davon ausgegangen, dass die Mehrzahl der Sicherheitsfunktionen von den Entwicklern ausgeführt wird. Die Prüfung auf Sicherheitslücken wird zentral für die gesamte Anwendung durchgeführt. Normalerweise übernehmen die für Qualitätssicherung oder Release-Engineering zuständigen Mitarbeiter diese Verantwortlichkeit, die die Grundlage dafür ist, dass dieses Modell problemlos in vorhandene Entwicklungsstrukturen integriert werden kann. Zeitpunkt und Häufigkeit von Analysen können im Hinblick auf flexibel festlegbare Testanforderungen gesteuert werden, von einem agilen Entwicklungsprozess bis zu einem stärker strukturierten Prozess (z. B. Entwicklung nach dem Wasserfallmodell).

Wie funktioniert es?

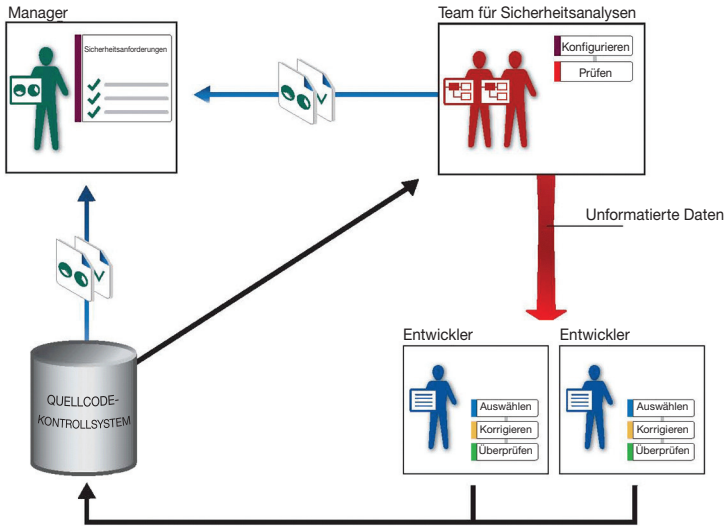
Die Sicherheitsanalyse kann entweder als Anforderung vor Beginn der Qualitätssicherung oder im Rahmen eines Abnahmetests nach Beginn der Qualitätssicherung durchgeführt werden. Die für Qualitätssicherung oder Release-Engineering zuständigen Mitarbeiter konfigurieren die Analyse entsprechend den zentralisierten Sicherheitsanforderungen, prüfen die gesamte Anwendung und leiten die Ergebnisse als Rohdaten an das Entwicklungsteam weiter. Anschließend sind die einzelnen Entwickler für die Analyse der Daten verantwortlich, um die kritischsten Schwachstellen zu ermitteln und geeignete Korrekturmaßnahmen durchzuführen. Bei dieser Aufteilung der Verantwortlichkeiten müssen die in der Entwicklung tätigen Mitarbeiter über umfassendes sicherheitsspezifisches Know-how und Fachwissen verfügen. Die für Qualitätssicherung oder Release-Engineering zuständigen Mitarbeiter übernehmen einfache Konfigurations- und Planungsaufgaben.

Je nach Größe der Anwendung und der bei der ersten Prüfung ermittelten Sicherheitslücken sind beim verteilten Modell möglicherweise mehrere Wiederholungen von Sicherheitstests erforderlich, um zu überprüfen, ob sich die Korrekturen als wirksam erweisen und die Funktionalität der Anwendung weiterhin gewährleistet ist. Sich wiederholende Tests sind auch bei der Entwicklung der Anwendung von entscheidender Bedeutung. Neu auftretende Abhängigkeiten und Interaktionen im Code können zu Sicherheitslücken führen, die bisher nicht ermittelt werden konnten. Durch häufigere Analysen erhöhen sich bei diesem Modell die Möglichkeiten für die Mitarbeiter, Schwachstellen frühzeitig zu erkennen. Allerdings muss hierbei ein vernünftiges Maß gefunden werden, um den Zeitplan für die Entwicklung nicht zu gefährden. Ein wichtiger Vorteil des verteilten Modells besteht darin, dass sich dieses Maß ermitteln lässt. Anpassungen können auf einfachere Weise an einem zentralen Scanprozess als an einzelnen Prüfungen wie beim unabhängigen Modell vorgenommen werden. Bei diesem Modell werden darüber hinaus redundante Abläufe vermieden, die dann auftreten, wenn die Entwickler für die Konfiguration und Ausführung eigener Analysen selbst verantwortlich sind.

Unternehmen, die Sicherheitslücken in Anwendungen vor deren Auslieferung oder Implementierung schließen, profitieren von enormen Kosteneinsparungen – intern und für Kunden, Geschäftspartner und andere Stakeholder, die mit ihrer Software arbeiten.
Seite 11

In welchen Fällen funktioniert es?

Für das verteilte Modell eignen sich am besten mittelgroße Entwicklungsteams, die nach einem formalen Prozess für die Softwareentwicklung arbeiten. Es bietet sich außerdem im Rahmen eines agilen Entwicklungsprozesses an, da sich durch die Häufigkeit von Tests die Chancen auf eine frühzeitige Erkennung und Beseitigung von Schwachstellen im Lebenszyklus erhöhen. Ein Entwicklungsprozess nach dem Wasserfallmodell bietet weniger Möglichkeiten, eignet sich aber ebenfalls für das verteilte Modell, da der Nutzen jeder Analyse durch den größeren Umfang von Produktmeilensteinen steigt.



Workflow:

Manager

Definition der Sicherheitsanforderungen

Mitarbeiter für Qualitätssicherung oder Release-Engineering

Konfiguration von Scannern für die Build-Integration

Synchronisation des Codes an jedem Meilenstein

Scannen von Meilensteinkomponenten

Bereitstellung von Rohdaten für die Entwicklungsabteilung

In Fällen, in denen der sicherheitsspezifische Kenntnisstand der Entwickler nicht ausreicht, können bei diesem Modell Experten hinzugezogen werden, um Ergebnisse aus der Analyse zu erhalten und die nötige Priorisierung durchzuführen. Möglicherweise ergeben sich geringfügige Nachteile, wenn die Mitarbeiter nicht mit der Anwendungsarchitektur vertraut sind, dennoch bietet dieses Konzept den Entwicklern genügend Spielraum, um sich auf ihre Verantwortlichkeiten bei der Codierung konzentrieren zu können. Dieses Konzept funktioniert nur, wenn die für die Sicherheitsprüfungen zuständigen Mitarbeiter Teil des Teams für die Softwareentwicklung sind. Das Modell muss im Rahmen des Lebenszyklus bei der Entwicklung und nicht als Teil eines gleichzeitig ablaufenden oder externen Prozesses umgesetzt werden.

In welchen Fällen funktioniert es nicht?

Das verteilte Modell eignet sich nicht gut für komplexe Anwendungen, große Entwicklungsteams oder Lebenszyklen in der Entwicklung, die nicht auf funktionalen oder komponentenbasierten Meilensteinen beruhen. Es funktioniert nicht mehr, sobald die Entwickler ihre Verantwortlichkeiten priorisieren müssen. Die Entwickler müssen dieselben Probleme wie beim unabhängigen Modell bewältigen, z. B. redundante Tätigkeiten und ein Mangel an detailliertem Fachwissen im Sicherheitsbereich und zur gesamten Anwendungsarchitektur. Wenn die kritischsten Sicherheitslücken während des Priorisierungsprozesses nicht präzise ermittelt werden oder einzelne Entwickler mit sich überschneidenden Codekomponenten arbeiten, ist der Produktivitätsverlust letztendlich größer als der Nutzen durch die Maßnahmen zur Beseitigung der Schwachstellen.

Unternehmen, die Sicherheitslücken in Anwendungen vor deren Auslieferung oder Implementierung schließen, profitieren von enormen Kosteneinsparungen – intern und für Kunden, Geschäftspartner und andere Stakeholder, die mit ihrer Software arbeiten.

Bewährte Verfahren

Die bestmöglichen Ergebnisse lassen sich beim verteilten Modell durch folgende Maßnahmen erreichen:

Beginnen Sie mit den Prüfungen möglichst frühzeitig im Lebenszyklus. Führen Sie die Analysen regelmäßig und auf der Grundlage eines festgelegten Zeitplans durch. Übertragen Sie den Entwicklern die Verantwortung für den Schutz einzelner Komponenten der Anwendung, um redundante Tätigkeiten weitestgehend zu vermeiden.

Benennen Sie einen sicherheitsbewussten Mitarbeiter des Teams als Mentor, damit dieser die anderen Entwickler bei Analysen, Priorisierungen und gegenseitigen Prüfungen von Fehlerkorrekturen anleitet.

III: Zentralisiertes Modell

Das zentralisierte Modell bietet das größte Maß an Flexibilität. Es kann in Teams mit beliebig vielen Mitarbeitern eingesetzt werden, unabhängig vom Prozess der Softwareentwicklung und der Komplexität der Anwendung. Das Modell empfiehlt sich aufgrund der anfänglichen Investitionen in die erforderlichen Ressourcen im Allgemeinen nicht als Einstieg für kleinere Teams. Programme für die Quellcodeanalyse, die zunächst mit einer der beiden anderen Methoden ausgeführt werden, entwickeln sich dank der Vorteile in puncto Effizienz und Messbarkeit der Ergebnisse letztendlich doch zum zentralisierten Modell. Dies gilt insbesondere bei anspruchsvolleren Anforderungen in der Anwendungsentwicklung und bei größeren Mitarbeiterteams.

Wie funktioniert es?

Im Gegensatz zu den beiden anderen Modellen, bei denen von den Entwicklern umfassendes sicherheitsspezifisches Know-how verlangt wird, können die Sicherheitsfunktionen beim zentralisierten Modell von der Gruppe ausgeführt werden, die am besten mit den Sicherheitslücken in der Software vertraut ist. Das für die Sicherheitsanalyse zuständige Team prüft die gesamte Anwendung und greift dabei auf eine zentrale Stelle für Know-how und Technologie zurück, unabhängig davon, ob diese im Lebenszyklus bei der Softwareentwicklung intern oder extern vorhanden ist. Die unbearbeiteten Ergebnisse werden von diesem Team priorisiert. Dank der hierbei zusammengestellten Informationen steht den Entwicklern ein mit Prioritäten versehener Ablauf zur Fehlerbehebung zur Verfügung, der auf der Dringlichkeit der Sicherheitslücken basiert. Die Wahrscheinlichkeit ist größer, dass die Mitarbeiter dieses Teams die Sicherheitsanforderungen bei der Konfiguration der Analyse richtig interpretieren, da sie die Problemstellungen auf Unternehmensebene beim Risikomanagement kennen.

Die Zuordnungen bei der Beseitigung von Sicherheitslücken werden kategorisiert und über ein System zur Fehlerverfolgung an die einzelnen Entwickler gesendet. Dadurch können alle Mitarbeiter des Teams den Fortschritt bei der Beseitigung von Sicherheitslücken verfolgen.

Entwickler haben so die Möglichkeit, sich stärker auf Weiterentwicklungen und Verbesserungen des Quellcodes zu konzentrieren, entweder durch Hinzufügen von Funktionen zur Software oder durch Umcodieren von vermeintlich fehlerhaften Elementen.

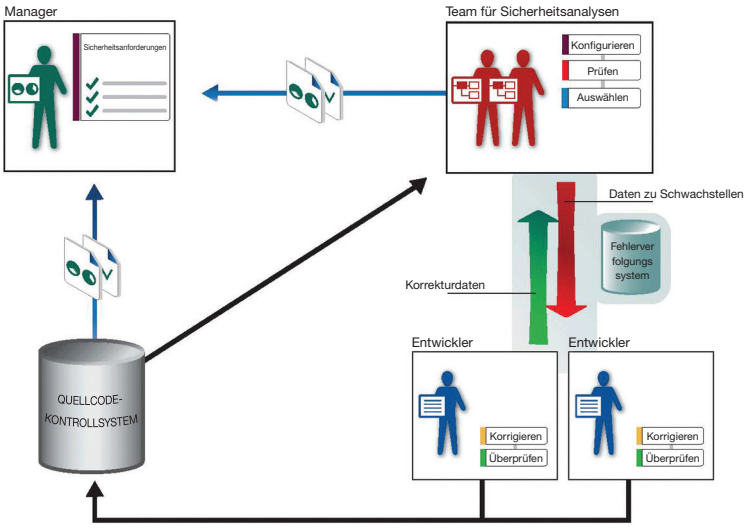
Die für Sicherheitsprüfungen zuständigen Mitarbeiter beraten die Entwickler bei diesem Modell im Hinblick auf kontextspezifische Korrekturmaßnahmen, sie stellen aber auch Richtlinien zur Fehlerbehebung entsprechend den unternehmensweiten Richtlinien zusammen. Beispiel: Ein für Sicherheitsprüfungen zuständiger Mitarbeiter stellt eine Sicherheitslücke durch eine SQL-Injection bei der Verarbeitung von Kreditkartennummern in einer Anwendung zur Onlineabrechnung fest. Die empfohlene Maßnahme zur Beseitigung dieser Sicherheitslücke kann lauten, den SQL-Code so zu ändern, dass eine gespeicherte Prozedur, eine parametrisierte SQL-Anweisung oder die unternehmensweite Standardroutine zur Prüfung von Kreditkarten verwendet wird. Das zentrale Sicherheitsteam kann diese Sicherheitslücke anschließend ordnungsgemäß dokumentieren und geeignete Maßnahmen basierend auf unternehmensspezifischen Richtlinien zuordnen, ohne dass die Entwickler voneinander unabhängige Entscheidungen treffen müssen.

Bei diesem Modell überprüfen die Entwickler selbst, ob ihre Maßnahmen zur Fehlerkorrektur wirksam waren. Das Ausmaß einer Sicherheitslücke wird nur durch die Analyse der gesamten Anwendung erkennbar. Es ist daher besser, wenn das Sicherheitsteam selbst für die Überprüfung verantwortlich ist.

In welchen Fällen funktioniert es?

Der wichtigste Vorteil des zentralisierten Modells besteht in der Flexibilität im Hinblick auf eine effiziente Integration – entweder im Rahmen des Lebenszyklus bei der Softwareentwicklung als Ergänzung für interne Mitarbeiter für die Softwareprüfung oder extern als Tool für Integratoren von Sicherheitslösungen oder Services zur Codeprüfung. Dieses Modell kann unabhängig von einem formalisierten Entwicklungsprozess eingesetzt werden, ohne klar definierte Struktur kann sich allerdings ein geringeres Maß an Effektivität ergeben. Durch die Zentralisierung von Konfiguration, Analyse und Priorisierung lässt sich dieses Modell am einfachsten verwalten. Es kann darüber hinaus innerhalb kurzer Zeit erweitert werden, wenn zusätzliche Mitarbeiter im Team hinzukommen oder sich der Projektumfang vergrößert.

Unternehmen, die Sicherheitslücken in Anwendungen vor deren Auslieferung oder Implementierung schließen, profitieren von enormen Kosteneinsparungen – intern und für Kunden, Geschäftspartner und andere Stakeholder, die mit ihrer Software arbeiten.
Seite 15



Workflow:

Manager

Definition der Sicherheitsanforderungen

Team für Sicherheitsanalysen

Konfiguration von Scannern für die Build-Integration

Abrufen des Codes für Analysen

Prüfen der gesamten Anwendung

Priorisierung der Ergebnisse

Zuordnen von Sicherheitslücken zu Entwicklern

Entwickler

Durchführung erforderlicher Korrekturen

Einchecken des Codes

Manager

Verfolgung von Fortschritten bei der Entwicklung, Prüfen der Daten zu Sicherheitslücken und Überwachen der Ergebnisse zur Fehlerbehebung

Aufgrund der verschiedenen Implementierungsszenarios, die bei Verwendung des zentralisierten Modells möglich sind (Entwicklung, interne Audits und externe Codeprüfung), sind häufig unterschiedliche Bereitstellungsmethoden für den Quellcode erforderlich. Die Verbindung zum Kontrollsystem für den Quellcode funktioniert problemlos, wenn die für die Analyse zuständigen Mitarbeiter intern auf den Lebenszyklus für die Softwareentwicklung zugreifen. Externe Mitarbeiten benötigen für Prüfungen Remotezugriff oder den Code auf Wechselspeichermedien, z. B. einer CD oder einem USB-Flash-Laufwerk.

In welchen Fällen funktioniert es nicht?

Das zentralisierte Modell kann für alle Projektszenarios angewendet werden, ist aber möglicherweise für kleine Teams oder Anwendungen unnötig kompliziert. Bei Verwendung des zentralisierten Modells muss die Unterstützung durch das Management gewährleistet sein, um sicherzustellen, dass genügend Mitarbeiter zur Verfügung gestellt werden. Es ist wichtig, dass für alle beteiligten Entwicklungs- und Sicherheitsabteilungen gemeinsame Ziele festgelegt werden. Ohne eine abteilungsübergreifende Zusammenarbeit und die Bereitstellung von genügend Mitarbeitern können Unternehmen keinen nachvollziehbaren Return on Investment erwarten.

Bewährte Verfahren

Die bestmöglichen Ergebnisse lassen sich beim zentralisierten Modell durch folgende Maßnahmen erreichen:

- Die Mitarbeiter der Sicherheits- und Entwicklungsabteilung müssen vor der Implementierung in die Planung und den Austausch von Informationen einbezogen werden, um eine gemeinsame Vereinbarung zu erreichen. In allen Teams muss zu folgenden zentralen Elementen Einvernehmen herrschen:
  - Anwendungsdesign und -architektur
  - Sicherheitsanforderungen
  - Sicherheitsschulungen
  - Priorisierung im Ablauf zur Fehlerbehebung
- Die für die Analyse zuständigen Mitarbeiter müssen angeleitet werden, damit sichergestellt ist, dass deren Ergebnisse, Priorisierung und Zuordnungen der Sicherheitslücken die größtmögliche Wirksamkeit haben.
- Der Prozess muss in vorhandene Technologien integriert werden (z. B. Systeme zur Fehlerverfolgung und integrierte Entwicklungsumgebungen), damit die Umstellung für die Mitarbeiter so reibungslos wie möglich abläuft und eine langfristige Effizienz gewährleistet ist.

Unternehmen, die Sicherheitslücken in Anwendungen vor deren Auslieferung oder Implementierung schließen, profitieren von enormen Kosteneinsparungen – intern und für Kunden, Geschäftspartner und andere Stakeholder, die mit ihrer Software arbeiten.

Auswahl des richtigen Modells

Alle drei Modelle stellen aktuelle Implementierungsszenarios dar, die bereits erfolgreich in mehreren Entwicklungsunternehmen eingesetzt wurden. Jeder Ansatz zeichnet sich durch einen bestimmten Grad an Flexibilität aus, mit dem spezielle Anforderungen in vorhandenen Prozessen erfüllt werden können. Die Modelle können letztendlich unterschiedlich gestaltet sein. Die grundlegenden Phasen und Funktionen sind allerdings eine erstklassige Anleitung für Unternehmen, die Sicherheitstests während der Entwicklung implementieren möchten.

Wenn Sie sich für eines der Modelle als Vorlage entscheiden, sollten die vorhandenen Ressourcen (sicherheitsspezifisches Know-how, Technologien und Servicepartner) und die Projektziele (weniger Sicherheitspatches, Wettbewerbsvorteile, Compliance) unbedingt zusammengestellt werden. In Anhang I wird ein hypothetischer Prozess zur Entscheidungsfindung und Implementierung auf der Grundlage von Faktoren vorgestellt, mit denen sich Entwicklungsunternehmen in der Praxis auseinandersetzen müssen.

Angesichts des immer stärkeren Bewusstseins von Sicherheitslücken in der Software als kritisches Problem bei der Informationssicherheit und der Verfügbarkeit von genauen und effizienten Technologien zur Analyse von Sicherheitslücken im Quellcode werden Sicherheitstests immer häufiger und immer erfolgreicher in die Softwareentwicklung integriert. Unternehmen, die Sicherheitslücken in Anwendungen vor deren Auslieferung oder Implementierung schließen, profitieren von enormen Kosteneinsparungen – intern und für Kunden, Geschäftspartner und andere Stakeholder, die mit ihrer Software arbeiten.

Voraussetzungen	Unabhängiges Modell	Verteiltes Modell	Zentralisiertes Modell
Effektive Reduzierung von Sicherheitslücken	X	X	X
Zentralisierte Reportings zu Sicherheitslücken und Fortschritten		X	X
Analyse in ein automatisiertes Buildsystem integriert		X	X
Unterstützung verteilter Entwicklungsteams	X	X	X
Übertragung der Fehlerbehebung an einzelne Entwickler			X
Möglichkeit zur Priorisierung des Ablaufs anhand bestimmter Kriterien			X
Erweiterbar auf 50 oder mehr Entwickler		X	X
Unterstützung komplexer Anwendungen			X

Anhang I: Fallstudie basierend auf dem zentralisierten Modell

Als Beispiel soll das Unternehmen Acme Software dienen, ein hypothetischer Anbieter einer webbasierten Anwendung für Rechnungstellung und Abwicklung, die auf dem Java™ 2 Enterprise Edition-Framework (J2EE) basiert. Am Hauptsitz des Unternehmens in Austin, Texas, arbeiten 25 Softwareentwickler (z. B. im Release-Engineering oder in der Qualitätssicherung). Weitere 25 Entwickler arbeiten im Rahmen einer Outsourcinglösung in Neu-Delhi, Indien. Die Entwickler arbeiten nach einem Agile-Prozess für die Softwareentwicklung mit mehreren Release- und Testphasen während des gesamten Lebenszyklus der Softwareentwicklung.

Das zentrale Produkt wurde in Java geschrieben und die Benutzerschnittstelle wurde als Web-Front-End basierend auf JavaServer Pages (JSP) entwickelt. Der gesamte Code ist zentral am Hauptsitz in Austin auf einem Standardsystem für die Überarbeitungskontrolle abgelegt. Der Buildprozess ist vollständig automatisiert und umfasst schrittweise Builds, die täglich bereitgestellt werden, sowie Builds mit dem gesamten Produkt, die jeweils morgens um 4:00 Uhr erstellt werden. Der aktuelle Code umfasst ca. 1,2 Mio. Codezeilen und kann sowohl auf Systemen mit Microsoft® Windows® als auch mit Linux® installiert werden.

Zur Verbesserung der Softwarequalität und zur Reduzierung der langfristigen Kosten im Sicherheitsbereich entschloss sich Acme, eine automatisierte Quellcodeanalyse in den vorhandenen Lebenszyklus bei der Softwareentwicklung zu integrieren. Als eine der wichtigsten Voraussetzungen benötigt Acme ein Testmodell, das die vorhandene komplexe Anwendung unterstützt und von einem dezentral arbeitenden und immer größer werdenden Team verwendet werden kann. Außerdem müssen Sicherheitsanalysen in das automatisierte Buildsystem integriert werden, um die Einheitlichkeit zu gewährleisten. Um alle Fortschritte und den Return on Investment zu verdeutlichen, müssen regelmäßig Berichte an das Management geliefert werden. Zudem möchten die Mitarbeiter alle Sicherheitslücken mit hoher Dringlichkeit beseitigen. Die für die Sicherheit zuständigen Mitarbeiter sind ebenso der Meinung, dass dies nur erreicht werden kann, wenn alle Ergebnisse und Abläufe effektiv priorisiert und einzelne Mitarbeiter mit Maßnahmen zur Fehlerbehebung betraut werden können. Im Folgenden sind die Möglichkeiten dargestellt, die jedes Modell zur Erfüllung dieser Anforderungen bietet.

Da genügend Mitarbeiter zur Verfügung stehen, um die gesetzten Ziele in puncto Softwaresicherheit zu erreichen, entscheidet sich Acme, das zentralisierte Modell umzusetzen, dass alle wichtigen Voraussetzungen erfüllt. Für die Analyse und Priorisierung von Sicherheitslücken wird ein zentrales Team für Sicherheitsanalysen zusammengestellt, in dem Release-Engineers, Mitarbeiter aus der Qualitätssicherung und mehrere Entwickler mit Know-how zu Sicherheitslösungen und zur Anwendungsarchitektur tätig werden. Für die Mitarbeiter im Sicherheitsbereich wurden folgende Aufgabenbereiche festgelegt:

Release-Engineering: Der Bereich Release-Engineering ist dafür verantwortlich, das Analysetool so zu konfigurieren, dass es Teil des automatisierten Buildprozesses ist, und alle Konfigurationsänderungen einzugeben, die nach Änderungen an der Anwendung während der gesamten Entwicklungsphase umgesetzt werden müssen.

Qualitätssicherung: Die Mitarbeiter für die Qualitätssicherung sind für die anfängliche Priorisierung verantwortlich. Acme entschloss sich, dass alle Sicherheitslücken mit hoher Dringlichkeit, die während eines Analysevorgangs protokolliert werden, umgehend von den Entwicklern bearbeitet werden müssen. Die Mitarbeiter für die Qualitätssicherung geben neue Fehler zu diesen Ergebnissen ein, bevor eine andere Priorisierung durchgeführt wird. Das Team ist darüber hinaus für die Prüfung von Fehlerkorrekturen verantwortlich, die von der Entwicklungsabteilung für vorherige Fehler bereitgestellt werden.

Unternehmen, die Sicherheitslücken in Anwendungen vor deren Auslieferung oder Implementierung schließen, profitieren von enormen Kosteneinsparungen – intern und für Kunden, Geschäftspartner und andere Stakeholder, die mit ihrer Software arbeiten.
Seite 20

Sicherheitsprüfung: Die für Sicherheitsaudits zuständigen Mitarbeiter sind für die Festlegung sicherer Codierungsrichtlinien und die Anpassung von Beurteilungsregeln für die Quellcodeanalyse verantwortlich. Sie übernehmen darüber hinaus die Priorisierung aller Ausnahmebedingungen, zweifelhaften Sicherheitslücken oder Schwachstellen im Design der Anwendung. Falls erforderlich, können die Mitarbeiter dieses Teams zu möglichen Sicherheitslücken, die beseitigt werden müssen, zusätzliche Fehler in das Fehlerverfolgungssystem eingeben.

Mit den sicherheitsspezifischen Verantwortlichkeiten, die unter den verschiedenen Gruppen in dem für die Sicherheitsanalysen zuständigen Team aufgeteilt wurden, entspricht der Ablauf weitestgehend dem Ablauf beim zentralisierten Modell.

- *Ein Build- oder Release-Engineer konfiguriert die Analyse auf Sicherheitslücken so, dass sie Teil des automatisierten Builds ist.*
- *Das für die Qualitätssicherung zuständige Team verwendet einen Basisfilter, der auf die Ergebnisse angewendet wird, um nur Sicherheitslücken mit hoher Dringlichkeit anzuzeigen.*
- *Ein Mitarbeiter des Sicherheitsteams mit Know-how zur Anwendungsarchitektur priorisiert die übrigen Ergebnisse, die nicht gefiltert wurden. Jeder Eintrag wird idealerweise nach der Priorisierung kategorisiert und dokumentiert, so dass künftig nur neue Einträge priorisiert werden.*
- *Die Entwickler beseitigen die ihnen über das Fehlerverfolgungssystem zugeteilten Sicherheitslücken. Sie müssen hierbei keine weitere Priorisierung vornehmen.*
- *Die Entwickler stellen den Code mit Fehlerberichtigungen, die von den für die Sicherheitsanalysen zuständigen Mitarbeiter geprüft wurden, erneut zur Verfügung. Der Zyklus beginnt damit erneut, wobei ein Release- oder Build-Engineer die Analyse möglicherweise rekonfigurieren muss, um alle zusätzlichen Prüfroutinen oder -filter, die nicht berichtigt werden konnten, aufzunehmen. Dies kann auch direkt durch den Engineer erfolgen.*
- *Das für die Qualitätssicherung zuständige Team überprüft die Sicherheitslücken, die als beseitigt gekennzeichnet wurden, und legt für alle weiteren Sicherheitslücken neue Fehler an. Das Sicherheitsteam priorisiert die neuen Ergebnisse, die auf mögliche Sicherheitslücken hinweisen.*

Unternehmen, die Sicherheitslücken in Anwendungen vor deren Auslieferung oder Implementierung schließen, profitieren von enormen Kosteneinsparungen – intern und für Kunden, Geschäftspartner und andere Stakeholder, die mit ihrer Software arbeiten.
Seite 21

Fazit

Dank der Aufteilung der Verantwortlichkeiten in diesem Modell sind die für die Qualitätssicherung zuständigen Mitarbeiter, die Sicherheitsanalysten und die Entwickler bei Acme in der Lage, sich auf die Bereiche zu konzentrieren, in denen sie ihr Fachwissen am besten einbringen können. Die für die Qualitätssicherung zuständigen Mitarbeiter identifizieren Sicherheitslücken mit hoher Dringlichkeit und ordnen sie mit Top-Priorität zur Fehlerbehebung zu. Die Sicherheitsanalysten können sich auf die Identifizierung von Sicherheitslücken im Anwendungsdesign und die Erarbeitung von Regeln zur Umsetzung der Unternehmensrichtlinien konzentrieren. Die Entwickler müssen sich nicht mehr um die Analyse von Ergebnissen oder die Priorisierung von Sicherheitslücken kümmern. Sie können sich vielmehr auf die Entwicklung der erforderlichen Fehlerberichtigungen im Code und das erneute Einchecken des Codes konzentrieren, damit das Sicherheitsteam diesen prüfen kann. Dank der Informationen, die den Entwicklern zusammen mit den ihnen zugeteilten Sicherheitslücken zur Verfügung gestellt werden, erweitern diese letztendlich ihr Know-how in Bezug auf die sichere Codierung in ihrem Arbeitsbereich. Entwickler, die mit der Datenbank arbeiten, schreiben beispielsweise qualitativ besseren Code zur Vermeidung von Sicherheitslücken durch SQL-Injections, sind aber keine Experten für Sessionmanagement oder Verschlüsselungen.

Auf der Grundlage des zentralisierten Modells ist Acme in der Lage, die dringendsten Sicherheitslücken zu ermitteln und äußerst effizient zu beseitigen. Regelmäßige Analysen und Überprüfungen an einem zentralen Standort führen zu einheitlichen Berichten über die Anzahl an Sicherheitslücken und die Fortschritte bei der Fehlerbehebung im Lauf der Zeit. Mit relativ wenigen Zyklen kann die Entwicklungsabteilung Managern, Geschäftspartnern und Kunden veranschaulichen, dass die Aufwände für das Testen des Quellcodes zu einer sichereren Software sowie zu einem Rückgang der laufenden Kosten für Wartung, Patches und mögliche Sicherheitslücken beiträgt.

Weitere Informationen

Wenn Sie mehr über die IBM Rational AppScan Source Edition erfahren möchten, wenden Sie sich bitte an den zuständigen IBM Vertriebsbeauftragten oder IBM Business Partner, oder besuchen Sie uns unter: **ibm.com**/software/rational/products/appscan/source/

Ryan Berg ist Senior Security Architect bei IBM. Er ist ein bekannter Referent, Schulungsleiter und Autor für Sicherheits-, Risikomanagement- und sichere Entwicklungsprozesse. Er hält in folgenden Bereichen Patente bzw. hat Patente angemeldet: Sicherheitsanalysen in mehreren Sprachen, Sicherheit auf Kernel-Ebene, Sprachen für zwischengeschaltete Sicherheitsanalysen und sichere Remote-Übertragungsprotokolle.

IBM Deutschland
IBM-Allee 1
D-71139 Ehningen
Germany
ibm.com/de

IBM Österreich
Obere Donaustraße 95
1020 Wien
ibm.com/at

IBM Schweiz
Vulkanstrasse 106
8010 Zürich
ibm.com/ch

Die IBM Homepage finden Sie unter: **ibm.com**

Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Stattdessen können andere, diesen funktional entsprechende Produkte verwendet werden.

Diese Veröffentlichung dient nur der allgemeinen Information.
Die in dieser Veröffentlichung enthaltenen Informationen können jederzeit ohne vorherige Ankündigung geändert werden. Aktuelle Informationen zu IBM Produkten und Services erhalten Sie bei der zuständigen IBM Verkaufsstelle oder dem zuständigen Reseller.

IBM erteilt keine Rechts- oder Steuerberatung und gibt keine Garantie bezüglich der Konformität von IBM Produkten oder Services mit den geltenden Gesetzen und gesetzlichen Bestimmungen. Der Kunde ist für die Einhaltung anwendbarer Sicherheitsvorschriften und sonstiger Vorschriften des nationalen und internationalen Rechts verantwortlich.

[1] „Studie: Flaw disclosure hurts software maker's stock,“ Robert Lemos, SecurityFocus (6.6.05).

[2] „Software Defect Reduction Top 10 List,“ B. Boehm und V. Basili, IEEE (01/2001).

[3] „Model Description Document,“ Systems Security Engineering, Capability Maturity Model Version 3.0 (15.6.03) http://www.sse-cmm.org/docs/ssecmmv3final.PDF

TAKE BACK CONTROL WITH **Rational.**

RAW14199-DEDE-00