# Durchgängige Sicherheit von Webanwendungen

IBM

Webanwendungen werden immer mehr zum Aushängeschild und zur zentralen Geschäftsplattform für Unternehmen. Doch bestehende Sicherheitsmaßnahmen auf der Netzwerkebene schützen nur unzureichend gegen gezielte Angriffe auf Schwachstellen in den Webanwendungen selbst. So tauchen in den Medien oft mehrmals wöchentlich Berichte über erfolgreiche Hackerattacken auf. Doch so einfach Sicherheitslücken ausgenutzt werden können, so einfach ist es für die betroffenen Unternehmen in der Regel auch, genau diese Lücken rechtzeitig zu erkennen und zu schließen.

## Schwachstellen in Webanwendungen und deren Auswirkungen

Moderne Webanwendungen sind nicht mehr mit den ersten Webseiten der frühen Internetzeit vergleichbar. Wurden früher häufig nur statische Inhalte mit wenig Interaktionsmöglichkeiten präsentiert, so sind die aktuellen Webanwendungen oft hoch interaktiv und durch diverse Technologien angereichert. So findet man z. B. Java, Javascript und AJAX oder flash-basierte Inhalte. Diese Technologien bieten den Vorteil, sehr dynamische Inhalte interessant darstellen zu können. Aus unterschiedlichen Quellen und Bereichen können Benutzern und Kunden Informationen über das Web-Frontend zugänglich gemacht werden. Das wachsende Interesse an Webanwendungen und an der Bereitstellung ebendieser bedeutet auch, dass die Entwickler diese Anwendungen erst einmal erstellen müssen. Es gibt inzwischen eine Vielzahl unterschiedlicher Plattformen und Frameworks um Webanwendungen zu erstellen. Auch die Anforderungen von Unternehmen an die Inhalte, sowie die Geschwindigkeit zur Erstellung der Webauftritte wachsen stetig. Aber dies bedeutet auch einen wachsenden Druck auf die Entwickler, die mehr Funktionalität und Performance in immer kürzer werdenden Releasezyklen unterbringen müssen. Die Webtechnologien werden immer komplexer und daher wird es für Unternehmen auch immer schwieriger, sie auf Sicherheit zu testen. Zudem sind Entwickler klassischerweise keine Sicherheitsexperten und die Forderung nach einer immer schnelleren Implementierung von immer mehr Funktionalität reduziert zusätzlich den Fokus auf den Aspekt Sicherheit. Doch je komplexer die Webanwendungen werden, desto mehr Sicherheitslücken enthalten sie und werden dadurch zu einem sehr interessanten und lohnenswerten Ziel für Hackerangriffe.

Auch schwerwiegende Sicherheitslücken können recht einfach gefunden und ausgenutzt werden. Ende des Jahres 2010 war es einem 16-Jährigen Schüler gelungen, in 17 Onlineauftritten von Banken Sicherheitslücken zu finden. Der Schüler war in der Lage, die Schwachstelle „Cross-Site Scripting“ (XSS) in den Seiten aufzudecken. Bei dieser Art Angriff schleust man manipulierten Script-Code auf Seiten ein und ist so in der Lage, zum Beispiel an sensible Daten betroffener Nutzer zu gelangen. Auch die häufig ausgenutzten Phishingattacken basieren auf dem Ansatz des Cross-Site Scriptings.

Wenn Sicherheitslücken an die Öffentlichkeit gelangen hat dies, neben negativer Presse, Imageverlust und Datenverlust, häufig auch rechtliche und regulatorische Konsequenzen für die Unternehmen. Denn wenn Hacker auf sensible Kundendaten zugreifen können, liegt in der Regel eine Verletzung von Compliancevorgaben vor, wie beispielsweise die Nicht-Einhaltung des Payment Card Industry Data Security Standards (PCI DSS) im Zusammenhang mit Kreditkartendaten. Diese Non-Compliance kann für Unternehmen schnell Strafen in Höhe von mehreren Hunderttausend Euro nach sich ziehen.

Aktuelle Untersuchungen des IBM X-Force Teams, das sich auf das Auffinden und Dokumentieren von Sicherheitslücken spezialisiert hat, belegen, dass etwa die Hälfte der derzeitigen Schwachstellen in Webanwendungen zu finden sind. Diese hohe Anfälligkeit machen Webauftritte für Kriminelle besonders interessant, da auch der Aufwand, die Schwachstellen auszunutzen, oftmals sehr gering ist.

## Sicherheitslücken identifizieren

Grundsätzlich gibt es zwei Vorgehensweisen, um Sicherheitslücken in Webanwendungen zu identifizieren, siehe Abbildung 1. Die statische Analyse des Anwendungs-Quellcodes (Whitebox-Tests) und die dynamische Analyse der Anwendung selbst in dem Zustand, wie diese auch letztendlich von den Anwendern genutzt wird (Blackbox-Tests). Beide Vorgehensweisen haben eine gemeinsame Teilmenge von auffindbaren Sicherheitslücken, aber auch eine Menge, die sich nur über die jeweilige Vorgehensweise selbst finden lässt.

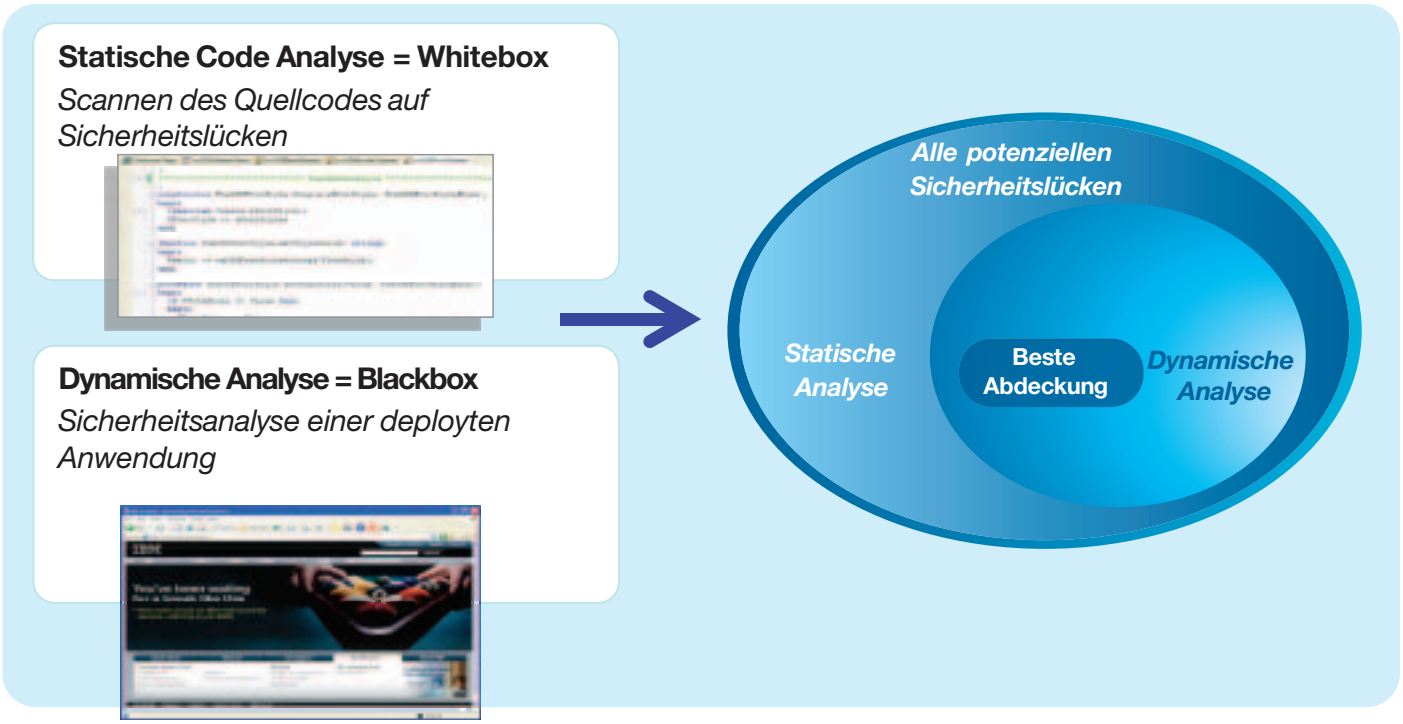


*Abbildung 1*: Sicherheitslücken mit statischer oder dynamischer Analyse aufdecken

## Blackbox-Tests (Web)

Der Blackbox-Test bezeichnet eine Methode des Softwaretests, bei der die Tests ohne Kenntnisse über die innere Funktionsweise und der verwendeten Technologie des zu testenden Systems entwickelt werden. Die Tests werden lediglich auf der Basis der Schnittstellen durchgeführt, die auch einem Nutzer der Anwendung zur Verfügung stehen. Im Kontext von Webanwendungen ist dies das HTTP/HTTPS Interface.

Das Aufspüren von Sicherheitslücken mittels des Blackbox-Tests lässt sich am besten als cleveres Erraten über manipulierte HTTP-Anfragen beschreiben. Dieses Erraten bzw. Herantasten an eine Schwachstelle macht unter anderem auch den Reiz für Hacker aus, die auf ähnliche Art vorgehen, siehe Abbildung 2. Der große Vorteil einer solchen Untersuchung liegt darin, dass man nahezu unabhängig von der verwendeten Technologie jede auf HTTP(S) basierende Anwendung mit den gleichen Tests prüfen kann. Die Tests betrachten beim Blackbox-Test üblicherweise das gesamte System, d. h. Server (Applikationsserver, Web-Server, DB-Server, etc.), externe Schnittstellen, Netzwerk, Firewalls, Drittanbieterkomponenten.

### Cross-Site Scripting – Blackbox-Test

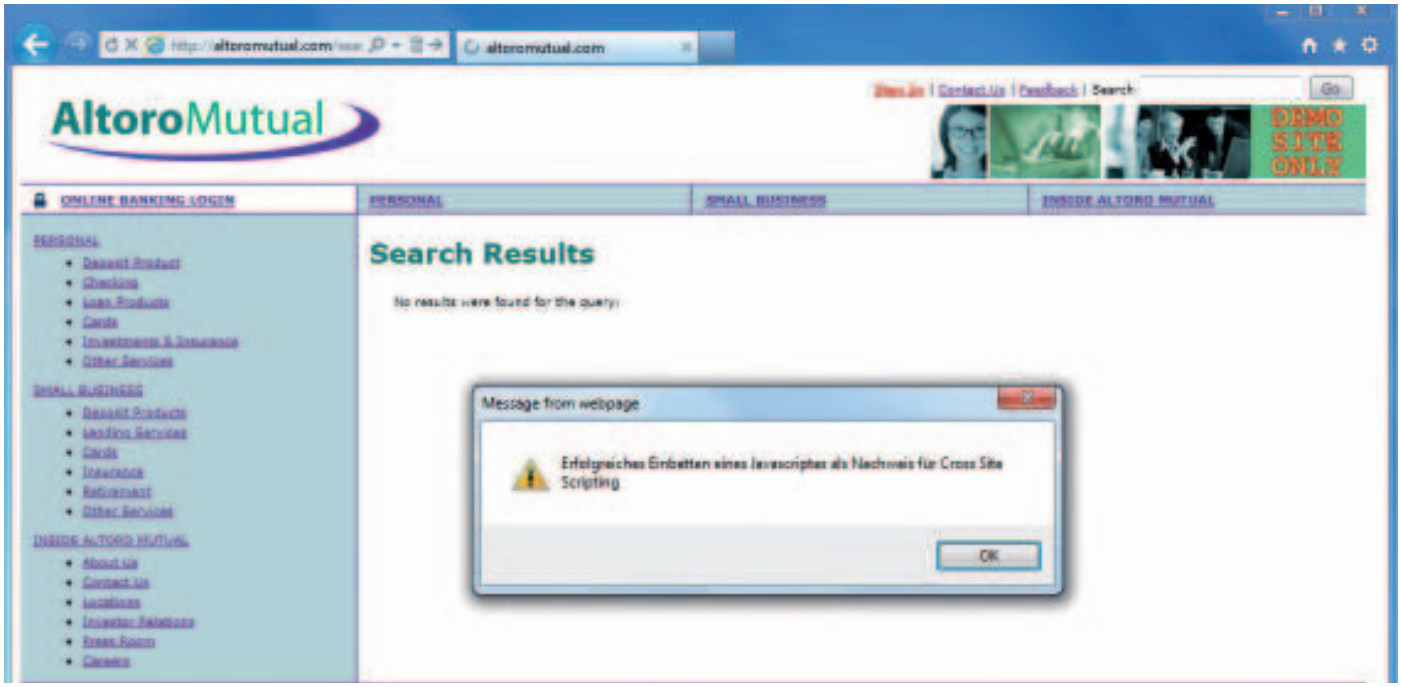


*Abbildung 2*: Blackbox-Test – Nachweis für Cross-Site Scripting

## Whitebox-Tests (Web)

Der Whitebox-Test bezeichnet eine Methode des Software-Tests, bei der die Tests mit Kenntnissen über die innere Funktionsweise des zu testenden Systems durchgeführt werden. Im Gegensatz zum Blackbox-Test wird direkt am Quellcode geprüft. Beispiele für einen Whitebox-Test können Kontrollfluss- oder Datenflussanalysen sein.

Das Aufspüren von Sicherheitslücken mittels des Whitebox-Tests lässt sich als Untersuchung einer unendlichen Anzahl von Verhalten in einem endlichen Ansatz, basierend auf dem Quellcode einer Anwendung, bezeichnen, was einem Code-Audit gleicht, siehe Abbildung 3. Um eine solche Untersuchung durchzuführen ist eine sehr gute Kenntnis der verwendeten Programmiersprachen und der eingesetzten Frameworks notwendig. Mit dieser Art der Untersuchung lassen sich Schwachstellen allerdings unabhängig von der Konfiguration finden. Eine große Herausforderung besteht darin, theoretische Schwachstellen, d. h. real nicht ausbeutbare Schwachstellen, zu eliminieren. Der große Vorteil dieser Vorgehensweise ist offensichtlich. Der Quellcode steht zur Verfügung und erlaubt, jedes mögliche Verhalten der Anwendung in jedem Winkel des Quellcodes nachzuvollziehen.

### Cross-Site Scripting – Whitebox-Test

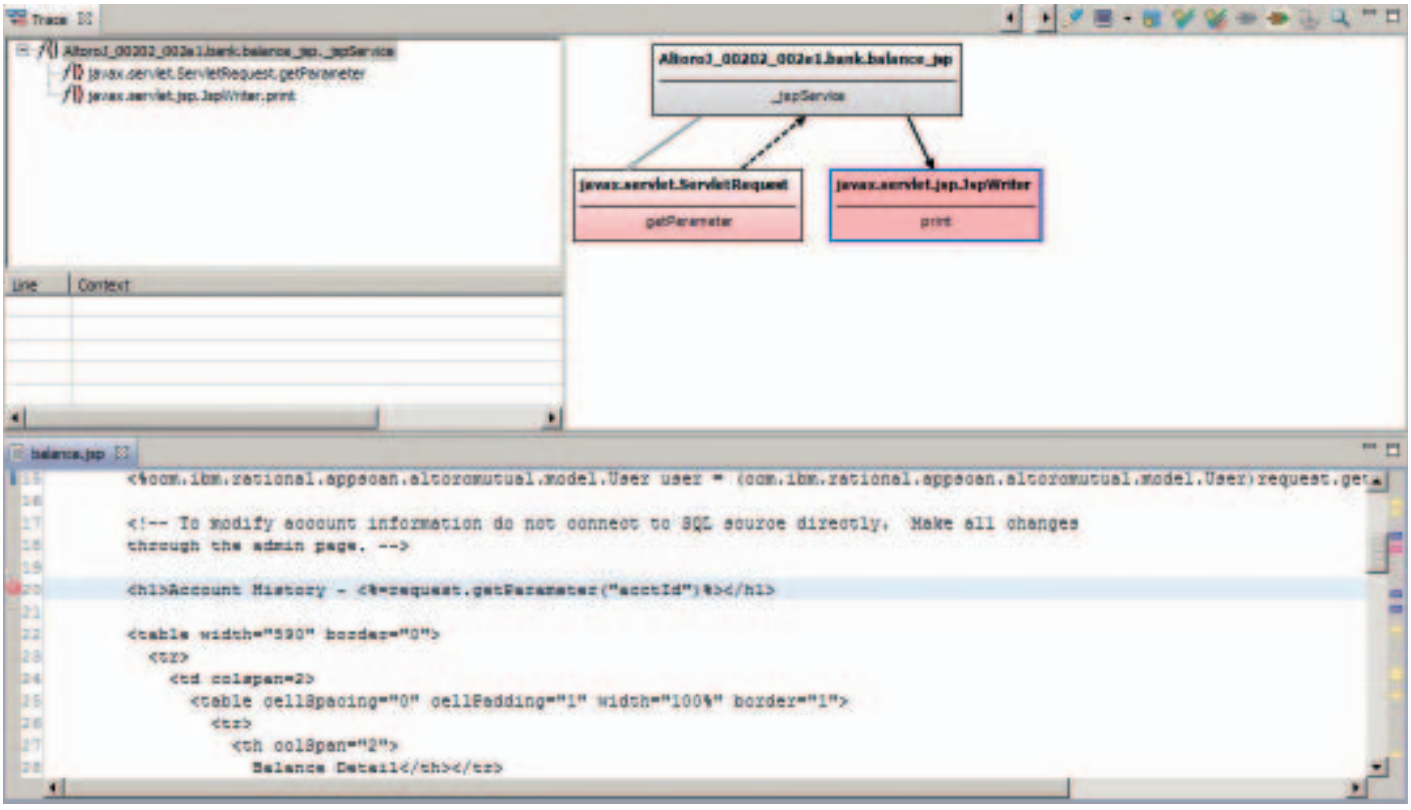


*Abbildung 3*: Whitebox-Test – Nachweis für Cross-Site Scripting

### Manuell oder automatisiert

Sowohl Blackbox-Tests, als auch Whitebox-Tests lassen sich auf manuelle und auf automatisierte Art und Weise durchführen. Ein Tool, welches eine Automatisierung ermöglicht, ersetzt selbstverständlich nicht den menschlichen Verstand. Der erlaubt es schließlich, Sicherheitslücken in der Logik von Abläufen innerhalb von Webanwendungen zu erkennen, wozu Algorithmen in Form von Programmen nur eingeschränkt fähig sind. Jedoch wird durch das breite Spektrum an Tests und deren schnelle Ausführung ein großes Maß an Tests mit maschineller Sorgfalt durchgeführt. Dies würde andernfalls tage- bzw. wochenlange Routinearbeit bedeuten, um eine gleichwertige Abdeckung an Sicherheitstests durch einen Menschen zu gewährleisten. Das automatische Generieren von Berichten über die gefundenen Schwachstellen erleichtert über den toolbasierten Ansatz die wohl mühevollste Arbeit eines manuellen Testers.

## Automatisierung von Blackbox-Tests

Ein Tool zur Automatisierung von Blackbox-Tests senkt die mit manuellen Sicherheitsüberprüfungen verbundenen Kosten, sowie deren zeitlichen Aufwand und hilft beim Schutz gegen Cyberangriffe, siehe Abbildung 4.

**Bei der Auswahl eines Tools für die Automatisierung von Blackbox-Tests ist zu beachten:**

- Testgenauigkeit: Das wichtigste Kriterium eines Blackbox-Testtools ist die Genauigkeit des Scanners, d. h. die Abstimmung von Tests auf die zu testende Anwendung. Dies wird beispielsweise durch eine genaue Analyse der Anwendung, der verwendeten Komponenten und Parameter und einer darauf basierenden Anpassung der Standardtests erreicht.
- Unterstützung von modernen und komplexen Technologien, wie JavaScript-Frameworks, URL-Rewriting, Flash, etc.
- Eine geringe False Positive Rate verringert den Aufwand für die Nachbearbeitung der Ergebnisse.
- Weitergehende Informationen über Schwachstellen und Integrationen. Dies betrifft insbesondere Erklärungen zu den durchgeführten Tests und den gefundenen Schwachstellen, detaillierte Korrekturhinweise für verschiedene Programmiersprachen und die Anbindung an Bugtracking-Systeme, um Fehler koordiniert zu adressieren.
- Berichte zur Einhaltung von Bestimmungen und Regularien wie: PCI Data Security Standard, Payment Applications Data Security (PA-DSS), ISO 27001 und ISO 27002 und Basel II.

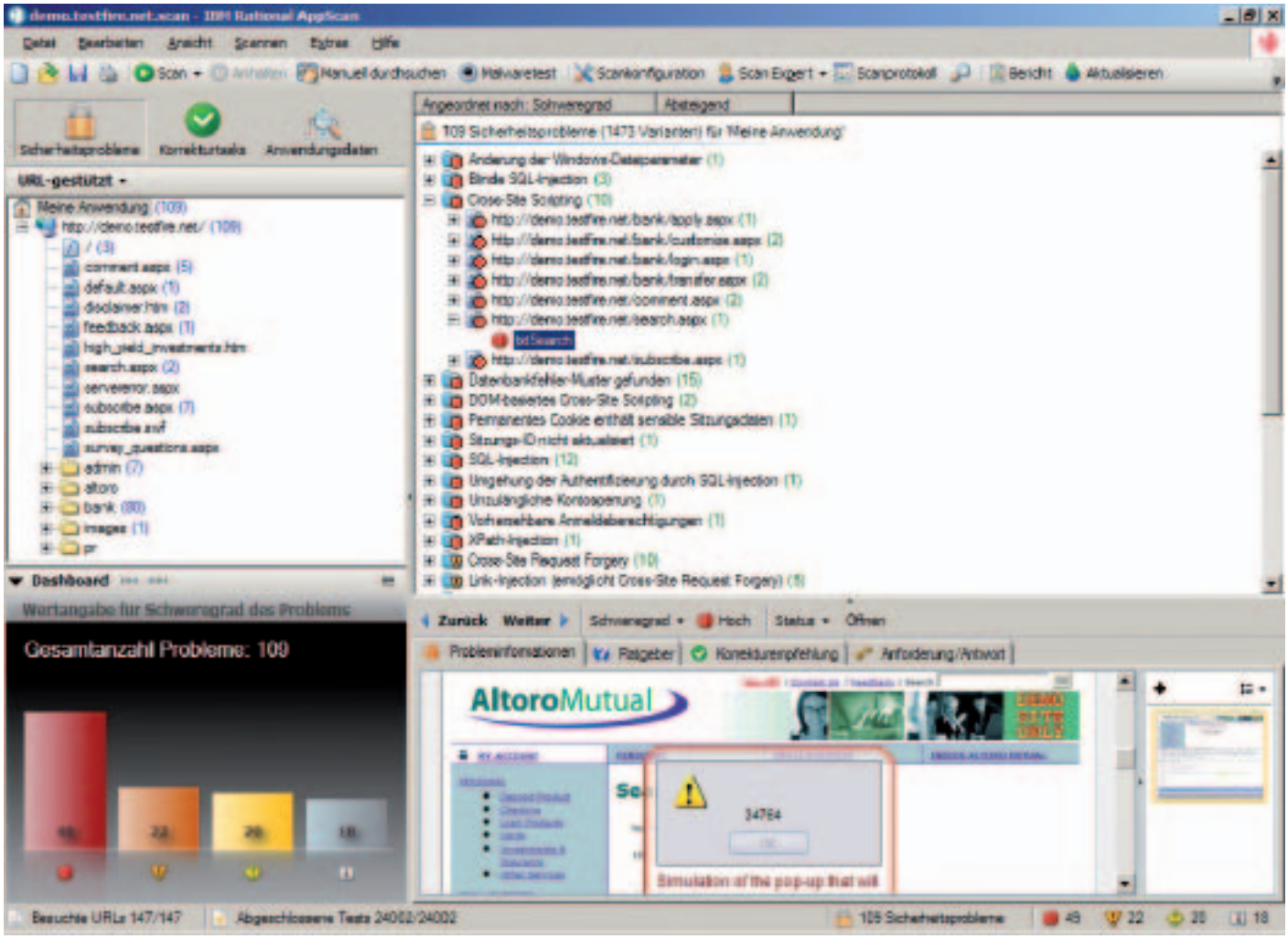


*Abbildung 4*: Blackbox-Testtool: AppScan Standard Edition

## Automatisierung von Whitebox-Tests

Ein Tool zur Automatisierung von Whitebox-Tests hilft Sicherheitsteams dabei, die Anwendungssicherheit zu verstärken, vertrauliche Daten zu schützen und die Einhaltung von Vorschriften zu verbessern, siehe Abbildung 5. Die Kombination dieses Quellcodetesttools mit der Sicherheitsüberprüfung von Webanwendungen bietet eine umfassende Abdeckung möglicher Sicherheitslücken.

**Bei der Auswahl eines Tools für die Automatisierung von Whitebox-Tests ist Folgendes zu beachten:**

- Unterstützung einer breiten Auswahl der größten und komplexesten Anwendungen in zahlreichen Programmiersprachen und deren Frameworks
- Vereinfachte Kooperation zwischen Sicherheit und Entwicklung durch das Angebot einer flexiblen Sichtungs- und Korrekturfunktion, die den Informationsfluss zwischen diesen Teams automatisiert
- Einbindung automatisierter Sicherheit in Ihre Entwicklungsprozesse durch die nahtlose Integration der Sicherheitscodeanalyse in Form einer automatischen Überprüfung während des Prozesses der Builderstellung
- Erstellen, Verteilen und Durchsetzen konsistenter Richtlinien und Unterstützung unternehmensweiter Metriken und Berichtsfunktionen in einer zentralisierten Richtlinien- und Analysedatenbank
- Identifizierung und Korrektur von Sicherheitsfehlern im Quellcode in frühen Stadien des Anwendungslebenszyklus durch die Integration in die IDEs der Entwickler

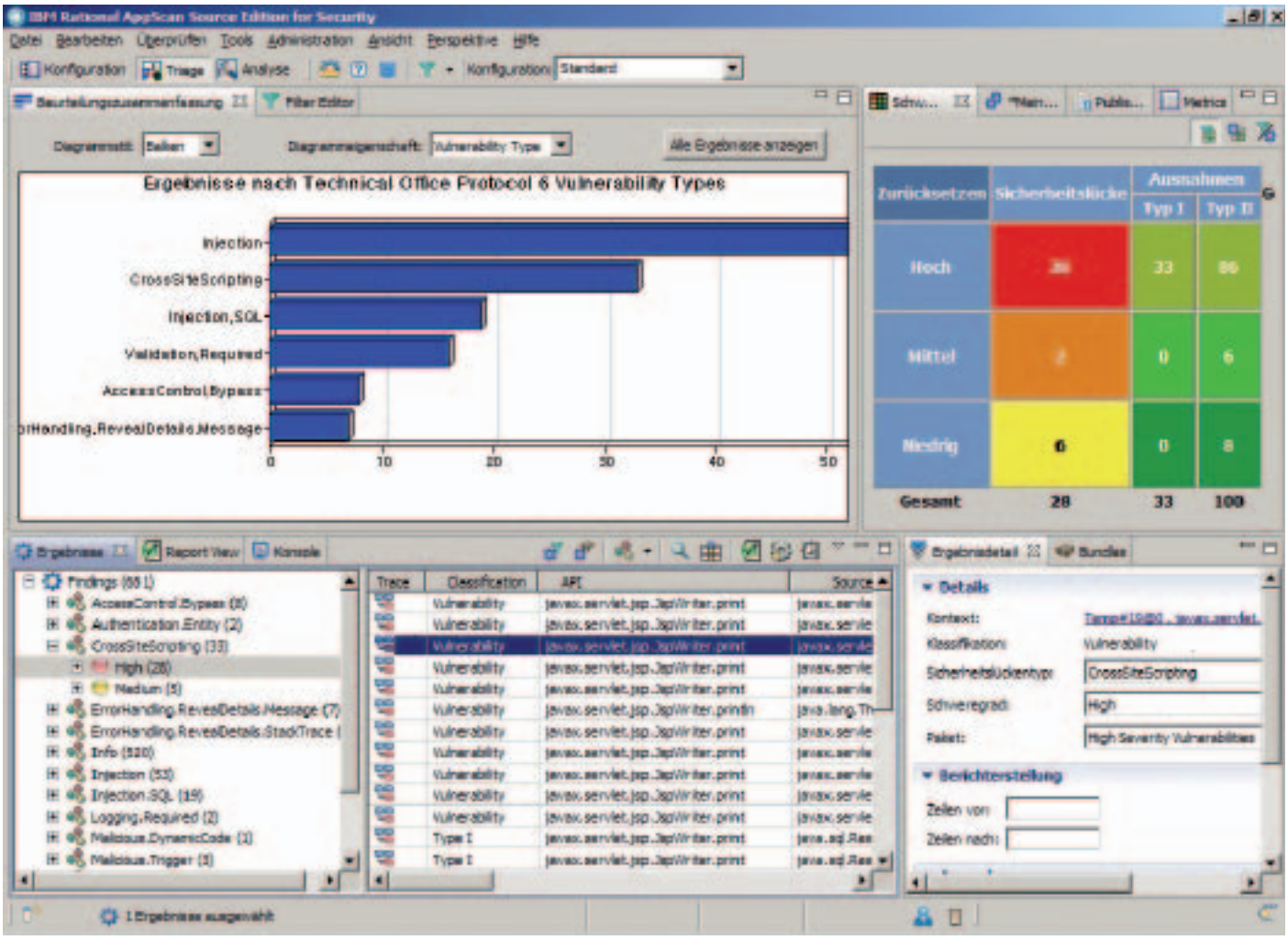


*Abbildung 5*: Whitebox-Testtool: AppScan Source Edition

## Sicherheit in der Softwareentwicklung – eine Prozessfrage

Es gibt viele Ansätze, wie Unternehmen, mit oder ohne Toolunterstützung, das Thema sichere Softwareentwicklung innerhalb ihrer Entwicklungsabteilungen etablieren. Wenige davon haben sich bewährt. Ein Fehler, der gerade bei der Einführung von Tools zur Überprüfung der eigenen Software hinsichtlich Schwachstellen oft gemacht wird, ist, diese dem Entwicklerteam zur Verfügung zu stellen, evtl. auch mit zugehörigen Schulungsmaßnahmen in der Hoffnung die Qualität der entwickelten Software zu erhöhen. Hieraus ergeben sich mehrere Probleme, u. a.:

- Redundant und unkoordiniert verrichtete Arbeiten
- Überforderung bei der Sichtung und Priorisierung der gefundenen Fehler
- Mangelndes Verständnis der Fehler oder mangelndes Verständnis der Behebungsmechanismen

Ein sich herauskristallisierter, funktionierender Ansatz kann wie folgt aussehen: Zunächst ist im Unternehmen eine Funktion erforderlich, die die Verantwortung für das Thema besitzt. Diese Person, in der Abbildung 6 vereinfacht „Manager“ genannt, hat die folgenden zentralen Aufgaben:

- Definition von Sicherheitsanforderungen sowie Sicherheitsrichtlinen und Überprüfung derer Umsetzung
- Beobachtung des Entwicklungsfortschritts der Anwendungen
- Einleitung von Weiterbildungsmaßnahmen, wo erforderlich
- Priorisierung der zu überprüfenden Anwendung

Zentrale Komponente dieses Ansatzes ist ein dediziert für das Thema verantwortliches, zentrales Sicherheitsteam. Dieses kann, je nach Größe des Unternehmens und der Anzahl der zu entwickelnden Anwendungen, von einem bis zu mehreren Mitarbeitern groß sein. Unternehmen, die sich ernsthaft mit dem Thema auseinandersetzen, scheuen nicht die Investition in mindestens eine solche Ressource. Häufig wird aber, um Kosten zu sparen, das Thema nur halbherzig angegangen. Aber diese Rolle ist kein Halbtagsjob.

Die Aufgaben des Sicherheitsteams sind u. a.:

- Vorbereitung der Anwendungen für die manuelle Überprüfung oder die automatisierte Überprüfung (z. B. im Rahmen des Build-Prozesses)
- Überprüfung der gesamten Anwendung
- Sichtung und Priorisierung der Ergebnisse
- Zuweisen von Schwachstellen an Entwickler, vorzugsweise über ein Bugtracking-System
- Center of Knowledge: Steht bei Rückfragen zum Verständnis von Schwachstellen oder Korrekturmaßnahmen zur Verfügung

Ist diese Arbeit getan, bleibt dem Entwickler nur noch die eigentliche Arbeit an seinem Teil des Anwendungs-Quellcodes. Ist der Fehler behoben, kann dieser noch optional einen Verifizierungsscan durchführen, um sicherzustellen, dass die Maßnahmen zur Fehlerbehebung greifen.

Bei größeren Organisationen kann noch eine Ebene zwischen dem Sicherheitsteam und dem Entwickler existieren, welche durch den Entwicklungsteamleiter besetzt wird. Dieser bekommt Schwachstellen vom Sicherheitsteam zugewiesen, die sich nur auf die von ihm verantwortete Anwendung beziehen. Diese Schwachstellen verteilt er dann an die verantwortlichen Entwickler.

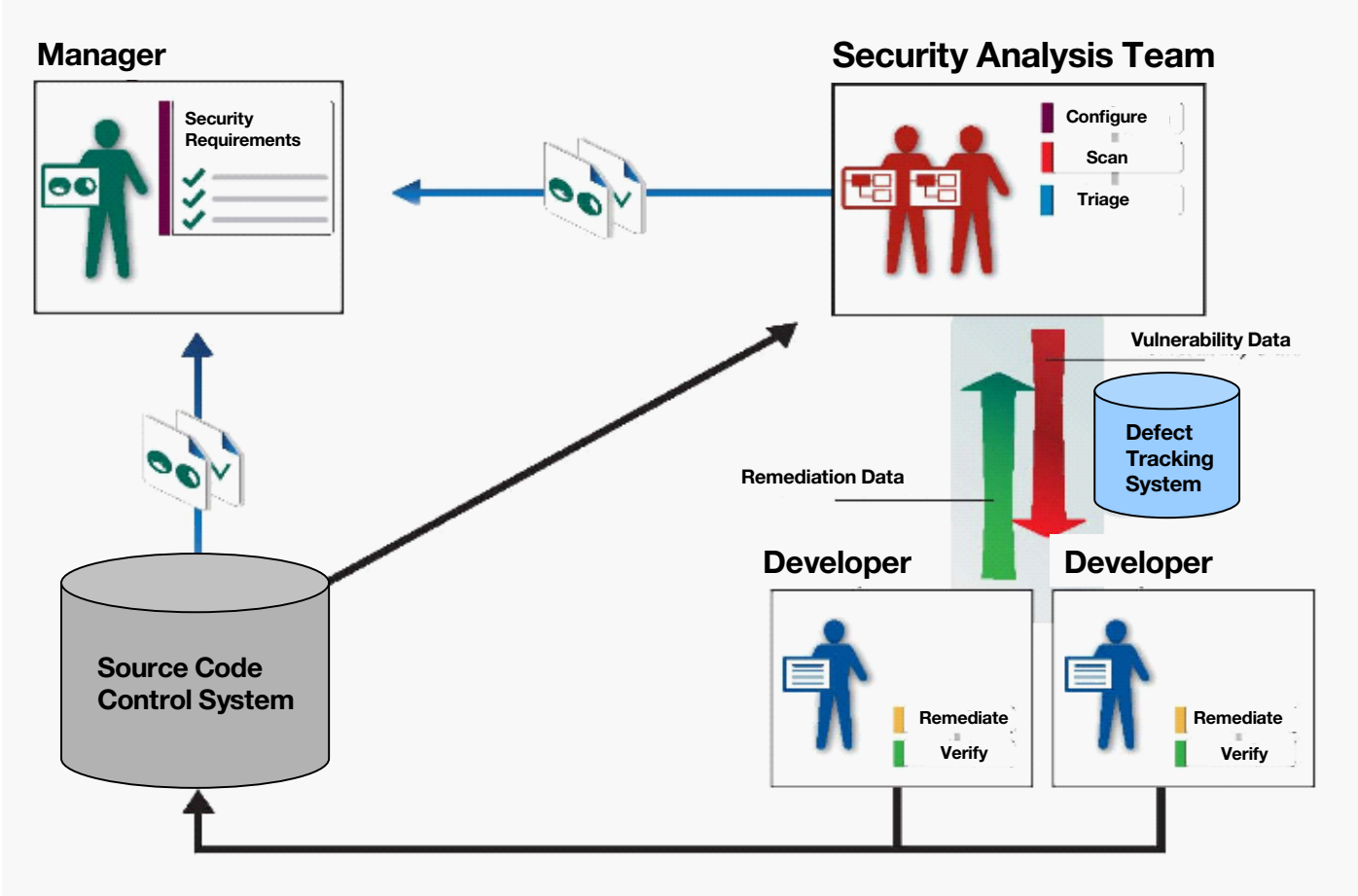


*Abbildung 6*: Sicherheitsprozess mit zentralem Sicherheitsteam

## Übergeordnetes Reporting und Schwachstellenkorrelierung

Genauso wichtig wie Schwachstellen zu identifizieren, ist es, diese Schwachstellen auch sauber zu dokumentieren und den entscheidenden Personen zugänglich zu machen. Denn es nützt einem Unternehmen wenig, wenn Schwachstellen zwar entdeckt werden, die entsprechenden Berichte aber nicht den Entwicklern zur Behebung zur Verfügung stehen. Daher ist ein übergeordnetes Reporting der Schwachstellen sinnvoll, z. B. über die Reporting Console. So können an einer Stelle zentral für alle Stakeholder wie Security Team, Entwicklung und Management stets die aktuellen Status der einzelnen Projekte sowie ein Gesamtüberblick eingesehen werden. Dies erlaubt es dem Security Team, die gefundenen Schwachstellen zu verteilen, den Entwicklern, auf die neuesten Bedrohungen zu reagieren, und dem Management, stets einen aktuellen Status und einen Überblick über die Schwachstellen zu haben.

Die Reporting Console bietet folgende Vorteile:

- Import und Darstellung sowohl von Blackbox-Testergebnissen (AppScan Standard) als auch Whitebox-Testergebnissen (AppScan Source) auf einer Plattform
- Hybride Analyse durch automatisierte Korrelierung der Ergebnisse von dynamischer und statischer Analyse, siehe Abbildung 7
- Performance-Metriken und Berichterstellung auf Unternehmensebene für eine gesamtheitliche Sicht auf Sicherheits- und Compliancerisiken
- Darstellung von Key Performance Indikatoren sowie Trendcharts um eine Messbarkeit von Veränderung für die Webanwendungen gewährleisten zu können
- Einheitliche Plattform für Sicherheitslücken zur Kommunikation und Kollaboration zwischen Security Team, Entwicklung und Management
- Rollen- und rechtebasierter Zugriff auf Scanergebnisse

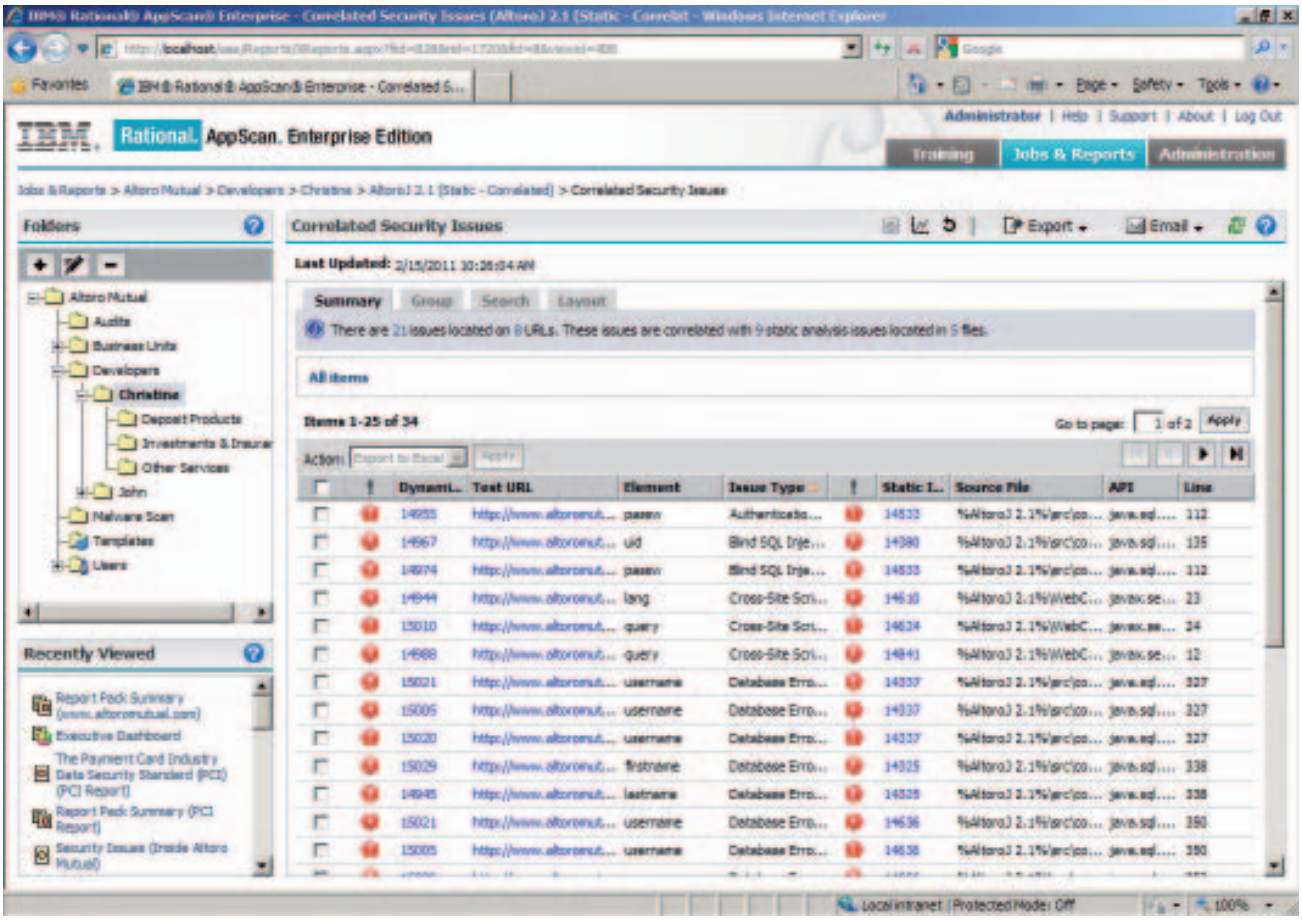


*Abbildung 7*: Reporting Console: Korrelierung von Ergebnissen aus statischer und dynamischer Analyse

Die Enterprise Edition umfasst die Fähigkeiten der Reporting Console und erweitert diese noch, um die Funktionalität Blackbox-Tests durchführen zu können. In Abbildung 8 ein Beispiel für das Management-Dashboard, um auf einen Blick zum Beispiel Trendcharts zu sehen. Bei Bedarf kann man vom Dashboard per Mausklick eine Ebene tiefer auf die Anwendungsberichterstellung gehen und, wenn gewünscht, bis auf einzelne Schwachstellen.


*Abbildung 8*: Enterprise Edition: Management-Dashboard mit Trendcharts

## Gesamtheitliche Betrachtung von Security

Da sowohl Blackbox-Tests, als auch Whitebox-Tests ihre Vor- und Nachteile haben, ist die Nutzung beider Technologien zu empfehlen. Und wenn beide Vorgehensweisen zudem Prozessintegrationen aufweisen, kann weiterer zusätzlicher Nutzen gezogen werden. So bringt z. B. ein übergeordnetes Reporting den Vorteil, Schwachstellen sowohl aus der dynamischen als auch der statischen Analyse darstellen zu können. Die sogenannte Korrelation der Schwachstellen erlaubt es, besser auf die gefunden Sicherheitslücken reagieren zu können. So sollte man z. B. eine erhöhte Priorität auf jene Lücken legen, die in beiden Analysetypen gefunden wurden und somit auch sehr wahrscheinlich eine reale Bedrohung für die Webseite darstellen.

Aber auch die Testautomatisierung und die Integration in den Buildprocess sind sinnvolle Ansätze. Hat sich erst einmal die Nutzung der Sicherheitstools etabliert, kann man sich wiederholende Aktionen automatisieren. So zum Beispiel die Scankonfiguration und den Start des Scanprozesses. Wenn der Scan nach dem erfolgreichen Build initiiert wird, kann danach der Securityexperte wie gewohnt die Scanergebnisse und Berichte sichten und bewerten. Bringt man dann als weiteren Baustein noch die Integration der Sicherheitstools in die Entwicklungsumgebung hinzu, wird das Gesamtbild abgerundet. Denn wenn Tools in bestehende IDEs der Entwickler eingebunden werden können, erhöht dies die Akzeptanz in der Entwicklungsabteilung, da man in der gewohnten Umgebung arbeiten kann.

Werden die genannten Maßnahmen effizient umgesetzt, dann amortisieren sich die Beschaffungskosten sehr oft innerhalb weniger Quartale. Doch die Nutzung einer Produktsuite von IBM hat noch weitere Vorteile. Die AppScan-Produkte arbeiten auch mit anderen Sicherheitsprodukten des Unternehmens zusammen. So wird unter anderem das sehr tiefgehende Sicherheits-Know-how des X-Force-Teams genutzt. Sobald die X-Force neue Sicherheitsbedrohungen für Webanwendungen identifiziert hat, werden diese an die AppScan-Entwicklungsteams weitergegeben und Tests schnellstmöglich implementiert. So ist es möglich, Kunden bereits kurz nach dem Auffinden der Schwachstelle ein Update für das Regelwerk zur Verfügung zu stellen.

Aber es gibt weitere Vorteile, speziell bei der dynamischen Analyse mit AppScan Standard. So wird beim Testen von Web-Services zur Generierung der Oberfläche aus der WSDL sowohl für in AppScan als auch im Rational Performance Tester der Generic Service Client (GSC) genutzt. Dies erlaubt, die Testabläufe für Web-Services für Sicherheit, Funktionalität und Performance in einem einheitlichen Client zu erstellen. Auch bietet Appscan Standard die Möglichkeit Scanergebnisse in den ISS SiteProtector zu exportieren. Der SiteProtector als Intrusion Prevention System kann die Scanergebnisse von AppScan nutzen und gezielt Angriffe auf Sicherheitslücken blockieren, die AppScan als anfällig identifiziert hat. Diesen Blockiermechanismus von Proventia nutzt man solange, bis man anhand der Fehlerbehebungsbeispiele von Appscan die Lücke im Anwendungsquellcode behoben hat.

## Fazit

Um wertvolle Informationen in Webanwendungen zu schützen, ist es wichtig, die Anwendungen durch deren kompletten Lebenszyklus zu testen; von der Entwicklung bis zum Betrieb. Sinnvoll ist es, sowohl dynamische Analysen als auch Sicherheitsüberprüfungen von Quellcodes durchzuführen, da beide unterschiedliche Typen von Lücken aufdecken können. Aber auch der gesamtheitliche Ansatz sollte berücksichtigt werden: IDE-Integration für Entwickler, übergeordnetes Reporting, sowie Automatisierung und Integration in den Buildprozess. Die effektivste Schutzmaßnahme gegen Sicherheitslücken in Anwendungen ist es, diese durch gut durchdachte Prozesse und gezielten Tooleinsatz gar nicht erst in Produktion gelangen zu lassen. Das Rational AppScan-Portfolio bietet all dies aus einer Hand, mehr dazu erfahren Sie unter

www-01.ibm.com/software/awdtools/appscan

## Über die Autoren


### Tobias Kutzer

ist seit 2007 im IBM Rational Technical Sales-Team in Deutschland. Sein Schwerpunkt innerhalb des Rational-Portfolios ist das Qualitätsmanagement, welches neben dem Testmanagement auch die Aspekte Functional, Performance und Security Testing umfasst.

E-Mail: tobias.kutzer@de.ibm.com

### Michael Kristen

ist IT-Spezialist und seit 2006 bei IBM Rational im Bereich Qualitätsmanagement mit dem Fokus auf Webanwendungssicherheit tätig.

E-Mail: michael.kristen@de.ibm.com

IBM Deutschland GmbH
IBM-Allee 1
71139 Ehningen
**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**



Bitte der Wiederverwertung zuführen

Rational. software

RAW14282-DEDE-00