

# Sichere Web-Anwendungen

## Vor- und Nachteile beim Einsatz von Sicherheitstestwerkzeugen

Markus Schumacher, Andreas Wiegenstein

*Schwachstellen in Web-basierten Applikationen haben vielfältige Ursachen. Automatische Testwerkzeuge können in kurzer Zeit große Applikationen auf eine Vielzahl von Fehlern überprüfen. Die heutige Technik reicht aber qualitativ noch lange nicht aus, um auf manuelle Tests durch Sicherheitsexperten verzichten zu können.*

### Einleitung

Mehr und mehr Unternehmensprozesse werden mit Web-basierten Anwendungen kontrolliert. Die Unternehmen erkennen zunehmend die Bedeutung von Applikationssicherheit für den Schutz ihrer kritischen Geschäftsprozesse. Allerdings kann bereits ein Sicherheitsleck in einer Web-Applikation einem Angreifer Zugang zu wichtigen Geschäftsdaten verschaffen. In falscher Sicherheit wiegt sich derjenige, der glaubt, dass hier Firewalls oder Verschlüsselungstechnologien helfen: viele Angriffspunkte entstehen auf logischer Ebene aufgrund von Schwachstellen in der Architektur, im Design oder in der Implementierung. Hinzu kommen lokale Anpassungen durch Administratoren sowie die parallele Installation verschiedener Anwendungen auf dem gleichen Server, durch die neue Verwundbarkeiten entstehen können.

Wie lässt sich dieses Problem am besten lösen? Sollten Sicherheitsexperten ins Haus geholt werden? Oder lässt sich dem Problem im Rahmen der üblichen Qualitätssicherung durch automatisierte Testläufe mit Testwerkzeugen beikommen? Die häufigsten Fragen in diesem Zusammenhang sind „Wie sicher ist unsere Software, nachdem wir ein Testwerkzeug zur Analyse darauf angesetzt haben?“ und „Wie ist die Kosten/Nutzen Bilanz beim Einsatz von Security Testwerkzeugen?“. Im vorliegenden Beitrag geben wir Antworten auf diese Fragen. Wir gehen zunächst auf die Zielsetzung von Sicherheitstests ein und zeigen dann grundsätzliche Unterschiede zwischen menschlichen Testern und der Methodik von Sicherheitstestwerkzeugen auf.

Sicherheitseigenschaften durch bestimmte Sicherheitsfunktionen gedacht. Ein typisches Beispiel ist der Einsatz von verschlüsselten Kommunikationsverbindungen, um die Vertraulichkeit der übertragenen Informationen zu sichern. Die Einhaltung solcher Sicherheitseigenschaften kann durch funktionale Tests nachgewiesen werden.

Ein anderer Aspekt ist allerdings die Sicherheit der Funktionalität von Anwendungen. So muss beispielsweise das Anlegen einer Bestellung oder die Verwaltung von Personaldaten sicher vor Manipulationen sein. Im Gegensatz zu den Sicherheitsfunktionen kann für Applikationen die Robustheit gegen Angriffe nicht auf herkömmliche Weise nachgewiesen werden.

Wie beim Testen im Allgemeinen gilt hier die Aussage von Edsger Dijkstra, dass Testen nur die Fehler aufzeigen kann, nicht aber die Fehlerfreiheit. Die Zielsetzung bei Sicherheitstests von Anwendungen ist analog hierzu, auf möglichst viele, typische Sicherheitsprobleme zu prüfen.

Da heutige Anwendungen in immer mehr sensitiven Bereichen, wie etwa bei Banken, in der Produktionssteuerung oder im Einzelhandel eingesetzt werden, besteht der Wunsch, potentiellen Sicherheitsproblemen möglichst effektiv auf den Grund zu gehen, um es einem möglichen Angreifer nicht zu leicht zu machen. Grundsätzlich können dazu menschliche Tester beauftragt oder Sicherheitstestwerkzeuge – so genannte „Application Security Scanner“ oder kurz „Scanner“ – eingesetzt werden. Im nächsten Abschnitt verdeutlichen wir prinzipielle Unterschiede zwischen der Testmethodik von Mensch und Scanner.

## 2 Mensch vs. Maschine

Um einige zentrale Unterschiede zwischen einem menschlichen Sicherheitstester und einem Testwerkzeug zu verdeutlichen,

## 1 Zielsetzung von Sicherheitstests

Wenn von Software-Sicherheit gesprochen wird, wird oft an die Gewährleistung von



Dr. Markus Schumacher

Geschäftsführer (CEO) der Virtual Forge GmbH. Themen: RO(S)I, Research, nachhaltige Sicherheit von

Applikationen.

E-Mail: markus.schumacher@virtualforge.de



Andreas Wiegenstein

Geschäftsführer (CTO) der Virtual Forge GmbH, Themen: Sicherheitsanalysen, Tools und Testmethoden.

E-Mail: andreas.wiegenstein@virtualforge.de

greifen wir Parallelen zum Schachspiel aus einem Vortrag von Rüdiger Weis auf. Er spricht von einem Schachspiel von Weltmeister Kramnik gegen das Schachprogramm Deep-Fritz. Interessant ist, dass Kramnik eigentlich nicht Schach gespielt hat. Er hat vielmehr gezielt versucht, die Algorithmen von Deep-Fritz zu überlisten. Kramnik hat in diesem Spiel vollkommen anders gespielt als gegen einen menschlichen Gegner.

Die Quintessenz des Vortrages von Weis ist, dass ein Schachcomputer im Vergleich zum Menschen zwar eine sehr große Rechenleistung hat, menschliche Spieler aber oft intuitive Entscheidungen treffen, die im Spiel entscheidend sein können und sich nur schwer oder gar nicht in Algorithmen abbilden lassen. Zusammenfassend sagt Weis, dass die Rechenleistung eines guten Schachprogramms in Kombination mit dem gesunden Menschenverstand eines nur mittelmäßigen Schachspielers nahezu unerschlagbar wäre.

Wir sehen hier eindeutige Parallelen zu Sicherheitstests von Anwendungen. Die Schwierigkeit besteht aufgrund der Komplexität in beiden Fällen darin, aus einer großen Menge von Alternativen die richtige Wahl zu treffen. Ein Scanner kann selbstständig große Codeteile und Applikationen schnell nach Mustern durchforsten, die sich algorithmisch darstellen lassen. Die logischen Zusammenhänge der einzelnen Applikationsaufrufe bleiben dem Scanner allerdings verschlossen.

Im Gegensatz dazu arbeitet ein menschlicher Experte intuitiv: er erkennt Eigenarten von Anwendungen und Quelltexten, die sich nicht algorithmisch beschreiben lassen. Wie ein potentieller Angreifer auch, muss der Tester dabei nicht den durch die Entwickler vorgedachten Regeln folgen, sondern kann kreativ vorgehen. Aufgrund seines Abstraktionsvermögens kann der menschliche Tester sowohl Inhalte als auch die Anwendungslogik und Testergebnisse verstehen, priorisieren und die Tests zur Laufzeit entsprechend anpassen. Der Mensch erkennt beispielsweise, dass ein bestimmtes Datenfeld eine Kreditkartennummer oder ein Passwort enthält und entsprechend geschützt sein müsste. Zudem kann der Mensch die Architektur und das Design einer Anwendung analysieren und so Assoziationen zwischen technisch nur lose verknüpften Prozessen erkennen und übergreifende Aspekte berücksichtigen.

Selbstverständlich gibt es auch Nachteile von menschlichen Sicherheitstestern. Anders als bei einem Scanner können sich die Ergebnisse einer Analyse deutlich unterscheiden, wenn verschiedene Tester involviert sind. Der Scanner wird immer zum selben Ergebnis kommen, Menschen hingegen wahrscheinlich unterschiedliche Resultate abliefern. Dies hängt einerseits von den Kenntnissen des Testers, andererseits von der individuellen Vorgehensweise ab. Ein weiterer Punkt sind die Kosten beim Einsatz von menschlichen Testern – auch wenn der Nutzen hoch ist, so ist es bei umfangreichen Anwendungen teuer, eine vollständige Testabdeckung zu erreichen.

Wir kommen daher zum Schluss, dass die Kombination eines flexibel konfigurierbaren Scanners mit einem erfahrenen Sicherheitstester, ähnlich wie im oben genannten Schach-Szenario, wesentlich effizienter ist, als beide Ansätze für sich alleine gesehen. Gemeint ist damit aber ausdrücklich nicht, dass der Tester den Bericht des Scanners interpretiert, sondern dass er unabhängig davon seinen individuellen Testansatz verfolgen muss.

Im Folgenden gehen wir ausführlicher auf die Eigenschaften von Scannern ein, um zu zeigen, welche Grenzen sie haben und warum menschliche Sicherheitstester und -methoden unverzichtbar sind.

## 3 Bewertungskriterien

Wir stellen zunächst pragmatische Bewertungskriterien für Scanner vor, die eine Software von Außen wie eine „Blackbox“ betrachten. „Blackbox“ bedeutet hier, dass der Scanner mit einer laufenden Anwendung interagiert und nicht deren Quellcode analysiert. Die Leistungsfähigkeit von Scannern kann dabei nach technischen Kriterien, aber auch nach wirtschaftlichen Kriterien beurteilt werden.

### 3.1 Technische Kriterien

Blackbox Scanner nutzen zunächst so genannte „Spidering“ Ansätze, um den gesamten Umfang einer Anwendung zu identifizieren. Dies ist sehr wichtig, da nur die Teile einer Anwendung auf Fehler getestet werden können, die das Tool auch gefunden hat. Der Scanner wertet dabei die Hyperlinks auf einer Seite aus und gelangt so zu weiteren Seiten. Ebenso müssen in diesem

Schritt Formularmasken korrekt ausgefüllt werden, um die Folgeseiten aufzuspielen. Alle gesammelten Links werden in einer Liste für die spätere Fehleranalyse abgelegt. Im besten Fall gelingt es einem Scanner, alle Seiten einer Anwendung durch seinen Algorithmus zu sammeln. Gerade komplexe Anwendungen erzeugen allerdings dynamische Seiteninhalte auf dem Client und enthalten sehr viele Formulare – beides Faktoren, die eine große Herausforderung für Spidering Algorithmen darstellen. Die Qualität von Spidering Algorithmen stellt daher ein wichtiges Bewertungskriterium dar.

Erst im zweiten Schritt wird dann jede (gefundene) Seite auf Sicherheitsprobleme untersucht. Die Eingabeparameter jeder Seite werden identifiziert und jede Seite wird dann mit Angriffscodes überflutet, um Sicherheitsfehler zu provozieren. Angriffscodes repräsentieren den Teil der Eingabe für eine Anwendung, der zur Ausnutzung einer Sicherheitslücke führen kann.

Ein guter Scanner muss „wissen“, wie sich typische Sicherheitsprobleme erkennen lassen. Er sollte einerseits alle Problemklassen kennen, andererseits die Ergebnisse eines Tests sinnvoll interpretieren. Die Qualität der Mustererkennung und die Interpretation der Testergebnisse sind daher weitere wichtige Bewertungskriterien. Ein Nebenaspekt sind dabei eingebaute Heuristiken, mit denen versucht wird, auf Varianten von Sicherheitsproblemen zu reagieren.

### 3.2 Kosten und Nutzen

Bei der Betrachtung der Gesamtkosten des Einsatzes eines Scanners müssen neben den Anschaffungskosten weitere Posten berücksichtigt werden. Dazu gehört zuerst der Aufwand, die Sicherheitstester im Umgang mit dem Werkzeug auszubilden. Zudem müssen Scanner richtig konfiguriert und an die Entwicklungs- und Testumgebung angepasst werden. Die Auswertung der Testergebnisse sowie die Kosten für die Beseitigung der gefundenen Fehler müssen ebenso berücksichtigt werden. Da Blackbox Scanner in der Regel erst spät im Entwicklungsprozess eingesetzt werden können – erst die lauffähige Anwendung kann getestet werden – sind diese Kosten entsprechend höher als bei Fehlern, die (von Menschen) zu einem früheren Entwicklungszeitpunkt erkannt werden.

Der Nutzen von Sicherheitstests ergibt sich dann, wenn die Testaufwände inklusive

der Schließung von identifizierten Sicherheitslücken geringer sind als die Kosten, die entstehen, wenn ein Sicherheitsproblem im schlimmsten Fall von einem Angreifer ausgenutzt werden kann. Im Schadensfall entstehen Kosten für den Notfallbetrieb, die Wiederherstellung des Normalbetriebs sowie Kosten durch Nutzungsausfall, Datenmanipulation und Datendiebstahl. Weitere nicht zu unterschätzende Faktoren sind beispielsweise der Verlust von Reputation (und damit Kunden), sowie Verstöße gegen Vorschriften zur aktiven Sicherung von IT Systemen (SOX, Basel II, HIPAA, etc.).

## 4 Stärken & Schwächen

Es gibt prinzipiell drei verschiedene Gründe, warum Scanner nicht alle Fehler finden: entweder ist die Bibliothek mit Angriffsmustern nicht umfangreich genug, oder ein erfolgreicher Angriff wird nicht als solcher erkannt, oder der Scanner findet eine mit Sicherheitslücken behaftete Seite nicht.

Um nun Stärken und Schwächen von Scannern zu erkennen wird zunächst eine Testumgebung benötigt, die es ermöglicht die Scanner im laufenden Betrieb zu beobachten. Dazu haben wir die Scanner auf eine präparierte Web-Applikation, die auf einem gehärteten Webserver läuft, angesetzt. So wurde ausgeschlossen, dass die Scanner Fehler des Webserver anzeigen. In die Testanwendung wurden 85 typische Schwachstellen unterschiedlicher Komplexität eingebaut. Jede dieser Schwachstellen wurde überwacht, d.h. es wurde protokolliert, ob der Angriffscodex, den der Scanner sendet, die Lücke erfolgreich ausnutzt oder nicht. Insgesamt haben wir sieben aktuelle Versionen von gängigen Scannern getestet (Stand: August 2006). Die Scanner wurden installiert und ohne Modifikation oder Erweiterung der mitgelieferten Angriffsmuster verwendet – so wie sie ein Nicht-Experte einsetzen würde.

Der Ablauf der Tests ist in der Abbildung auf dieser Seite dargestellt. Im ersten Schritt greift der Scanner auf die Test-Applikation zu (1). Er interpretiert die Webseite und versucht mit Hilfe eines Spidering Algorithmus allen Links zu folgen. Jede der so gefundenen Seiten enthält jeweils eine überwachte Schwachstelle. Diese versucht er durch verschiedene Angriffsvektoren auszunutzen (2). Es wird dabei protokolliert, auf welche der präparierten Schwach-

stellen der Scanner zugreift, mit welchen Angriffsmustern er arbeitet und ob er einen Angriff erfolgreich durchführen konnte (3,4). Ein erfolgreicher Angriff heißt aber nicht notwendigerweise, dass dies auch vom Scanner bemerkt wird. Erkennt der Scanner einen durch ihn erfolgreich durchgeführten Angriff, protokolliert er dies in seinem Abschlussreport (5).

Nach jedem Testlauf wurde dieser Abschlussreport mit der Protokollierung der Test-Applikation verglichen. Durch die Differenzen ließ sich schnell erkennen, welche Stärken und Schwächen der untersuchte Scanner hat. Im Folgenden sind die wichtigsten Ergebnisse dargestellt.

Zunächst ist festzustellen, dass Scanner sehr schnell arbeiten und automatisch gestartet werden können – auch außerhalb der Bürozeiten. Dies ist insbesondere vorteilhaft, wenn Unternehmen sehr häufig umfangreiche Tests durchführen müssen.

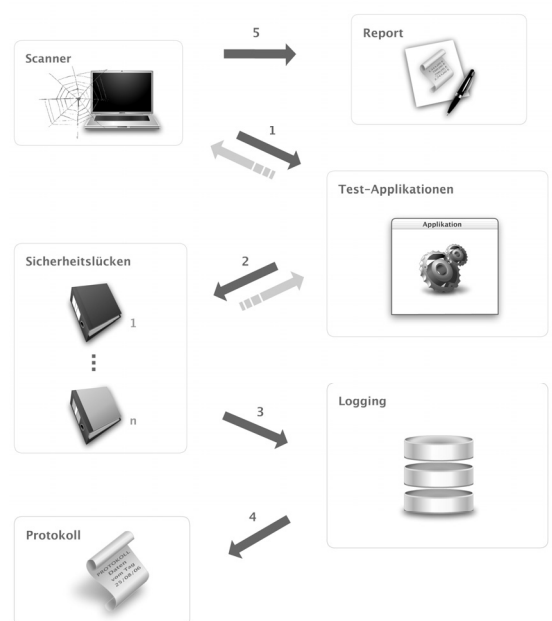
Die Scanner haben auch viele der einfachen Testfälle unterschiedlicher Fehlerarten in unserem System erkannt, im Bericht detailliert beschrieben und überwiegend korrekt bewertet, so dass eine Priorisierung bei der Fehlerbehebung möglich ist. Allerdings wurde häufig derselbe Fehler mehrfach als Problem angezeigt. Ein Bericht wird hierdurch unnötig umfangreich und bindet somit mehr Ressourcen bei der Auswertung und bei der Umsetzung von Maßnahmen, da für jede Fehlermeldung erkannt werden muss, dass es sich hier um einen bekannten Fehler handelt.

Unabhängig davon eignet sich ein Scanner sehr gut, um automatisierte Regressionstests durchzuführen, die eine korrekte Behebung der (vom Scanner) gefundenen Fehler sicherstellen.

## 5 Technische Defizite

### 5.1 Spidering

Wie bereits festgestellt muss ein Scanner zunächst eine verlinkte Seite finden, bevor er sie analysieren kann. Hierzu haben wir verschiedene Verlinkungstechniken als Testfälle definiert. Bemerkenswert ist, dass einer der getesteten Scanner keinen der



Spidering-Testfälle gelöst hat. Auch von den übrigen Scannern konnte maximal die Hälfte der Testfälle im Bereich Spidering geknackt werden. Hier führt auch eine Anpassung der Konfiguration des Scanners nicht zu einer Verbesserung, da die Spidering-Algorithmen nicht beeinflussbar sind.

Auch das automatische Ausfüllen von Formularen in der Anwendung bereitete den Scannern größere Schwierigkeiten, so dass viele weiterführende Seiten nicht gefunden wurden. Manche Scanner bieten allerdings Formulartabellen an, um beim Auftreten von bestimmten Parametern einen hinterlegten Wert einzusetzen, wie etwa Authentisierungsdaten. Im Schnitt wurde nur etwa die Hälfte aller Formulare korrekt ausgefüllt, so dass der Scanner die Folgeseiten erreichen konnte.

### 5.2 Erkennungsrate

Die Erkennungsrate von korrekt identifizierten Schwachstellen lag in unserem Test durchschnittlich bei lediglich acht von 85 Fehlern; der beste Scanner hat 14 Schwachstellen gefunden und das schlechteste Ergebnis lag bei vier. Interessanterweise haben einige Scanner mehrfach zwar korrekte Angriffscodes an das System geschickt, konnten dies aber selbst nicht erkennen (so genannte *False Negatives*). Diesbezüglich hat der beste Scanner bei nur vier Schwachstellen nicht erkannt dass er sie eigentlich geknackt hatte, der schlechteste hingegen bei 27, wobei der Durchschnitt bei 11 Fehlinterpretationen lag. Ein

interessantes Phänomen war auch, dass ein Scanner zu jeder Schwachstelle seine „erfolgreichen“ Angriffscodes dargestellt hat, diese aber nicht nachvollziehbar waren: bei manueller Eingabe der manipulierten Angriffs-URL wurde der Fehler daher nicht ausgelöst, was beim Tester darauf schließen lassen könnte, dass der Scanner ein „False Positive“ gemeldet hat.

Hinsichtlich der tatsächlich entstandenen „False Positives“ (also Fehler, die gar keine sind) waren die Testergebnisse recht unterschiedlich. Der Bestwert lag bei nur einem False Positive, der schlechteste Wert bei 29 und im Mittel waren es 11 – es wurden also insgesamt mehr False Positives als tatsächliche Schwachstellen gemeldet. Die folgende Tabelle fasst die Ergebnisse unseres Tests zusammen:

	Min	Max	AVG
Erkennungsrate	4	14	8
False Negatives	4	27	11
False Positives	1	29	11

Der Aufwand, dies im Einzelnen nachzuprüfen, ist sehr hoch. Es wurden zudem bestimmte False Positives auch noch mit bekannten Fehlern bestimmter Produkte verwechselt, die gar nicht installiert waren (z.B. „Extropia WebBanner Remote Command Execution“). Da dies bei mehreren Scannern in gleicher Weise aufgetreten ist, kann davon ausgegangen werden, dass hier verschiedene Scanner zumindest teilweise die gleichen Regelwerke verwenden.

Im Ergebnis ist hier festzuhalten, dass mit den getesteten Scannern zum Einen die nötige Testabdeckung nicht gewährleistet werden kann und zum Anderen die Ausführung unzureichend ist. Ein Tester, der im Anschluss an einen automatischen Scan eine Bewertung durchführt, darf nicht davon ausgehen, dass alle Vorkommen eines bestimmten Fehlers gefunden wurden. Er muss daher komplett von vorne beginnen.

## 6 Logische Schwachstellen

Hinsichtlich logischer Defizite gibt es zwei generelle Problemzonen: Scanner verstehen einerseits die Semantik der Daten nicht, die von einer Anwendung verarbeitet werden und sie können andererseits keine Aussagen darüber treffen, ob ein Benutzer die erforderliche Berechtigung hat eine bestimmte Aktion auszuführen. Werden also bei-

spielsweise vertrauliche Daten eines Benutzers über eine ID referenziert, so kommt der Scanner (im Gegensatz zum Menschen) nicht auf die Idee, diese ID zu ändern und zu prüfen, ob nun vertrauliche Daten eines anderen Benutzers erscheinen. Wie zu erwarten war, haben die Tests gezeigt, dass kein einziger Scanner einen der entsprechenden Testfälle in diesem Bereich erkannt hat. So wurden beispielsweise von keinem Scanner Probleme in den Bereichen Zugriffskontrolle, Anwendungslogik, Forceful Browsing, In-Band Signalling und File-Upload gefunden. Für komplexe Geschäftsanwendungen (wie z.B. Applikationen, die auf SAP Netweaver oder IBM WebSphere, aufsetzen) sind Scanner daher aus unserer Sicht nicht geeignet.

## 7 Generelle Defizite

Zum dritten Bereich, den generellen Defiziten von Scannern, zählt, dass diese prinzipiell keine Design- oder Architekturfehler finden können, keine allgemeinen Risiken einer Anwendung erkennen und somit in ihren Berichten auch keine sinnvollen Angaben über ihre Testabdeckung machen, an die ein Mensch dann anknüpfen könnte. Zudem sind die Berichte teilweise sehr umfangreich, was die Auswertung erschwert. So wurde in einem Fall das Anzeigen der IP Adresse in Fehlermeldungen bei einem Scanner gleich 7.735 (!) mal gemeldet, und dies sogar an zwei verschiedenen Stellen im Bericht. Eine Aggregation dieser Meldung wäre wünschenswert gewesen. Die Spitzenreiter in unserem Test haben die (wenigen) gefundenen Schwachstellen auf 946 bzw. 742 Seiten dokumentiert.

## Fazit

Scanner eignen sich sehr gut, um „Low Hanging Fruits“ effektiv zu identifizieren und auch in Regressionstests zu überprüfen. Der Aufwand hierfür ist allerdings nicht zu unterschätzen – wie der Test gezeigt hat, wird nur ein sehr gut konfigurierter Scanner halbwegs brauchbare Ergebnisse liefern. Dennoch sind die Erkennungsraten recht niedrig und die False Positive Raten zu hoch. Um ein wirklich hohes Sicherheitsniveau zu erreichen, ist die Kreativität und das Know-How eines menschlichen Sicherheitstesters unverzichtbar – nur so können logische Schwachstellen gefunden und die generellen Defizite eines Scanners ausgeglichen werden.

Scanner können zwar große Bereiche einer Applikation viel schneller als ein Mensch untersuchen, doch die Erkennung des einzelnen Fehlers gelingt dem Menschen wesentlich besser als der Maschine. Um somit zu unserer Eingangsfrage „Wie sicher ist unsere Software, nachdem wir einen Scanner zur Analyse darauf angesetzt haben?“ zurückzukehren: Der Scanner kann technisch bedingt nicht alle Fehler in einer Anwendung finden. Menschliche Experten mit dem entsprechenden Know-How aber schon, sofern genug Zeit zur Verfügung steht. Es ist eine Frage des Risikomanagements zu entscheiden, wer die nach einem Scannerlauf verbleibenden Fehler finden soll – der Sicherheitsexperte oder der Hacker?

## Literatur

- [1] Mark Curphey, Rudolph Araujo, *Web Application Security Assessment Tools*, IEEE Security & Privacy (Vol. 4, No. 4), July/August 2006
- [2] Arian Evans, *Software Security Quality: Testing Taxonomy and Testing Tool Classification*, 2nd Annual OWASP Conference, Washington DC, Oct. 2005
- [3] Gary McGraw, *Software Security: Building Security In*, Addison-Wesley Professional, 2006
- [4] Michael Howard, David LeBlanc, and John Viega, *19 Deadly Sins of Software Security – Programming Flaws and How to Fix Them*, Osborne McGraw-Hill, 2005
- [5] Sverre H. Huseby, *Innocent Code*, John Wiley & Sons, 2003
- [6] Gary McGraw, *Software Security: Building Security In*, Addison-Wesley Professional, 2006
- [7] H.Q. Nguyen, B. Johnson, M. Hackett, *Testing Applications on the Web*, Wiley Publishing, 2003
- [8] Holger Peine, *Stefan Mandel: Sicherheitsprüfungswerkzeuge für Web-Anwendungen*, Technical Report, FHG IESE, 2006
- [9] Jeffrey Rubin, *Review: Web Vulnerability Scanners*, Secure Enterprise Magazine, Sept. 2006
- [10] Jan Steffan, *Markus Schumacher, Collaborative Attack Modeling*, 17th ACM Symposium on Applied Computing (SAC 2002), Special Track on Computer Security, Madrid, Spain, March 2002
- [11] Rüdiger Weis, *Künstliche Unintelligenz*, Chaos Communication Congress (19C3), 2002 <http://ftp.ccc.de/congress/2002/lectures/19C3-414-kuenstliche-unintelligenz.mp3>
- [12] A. Wiegstein et al., *Web Application Security Scanners – a Benchmark*, Whitepaper, 2006 [http://www.virtualforge.de/whitepapers/web\\_scanner\\_benchmark.pdf](http://www.virtualforge.de/whitepapers/web_scanner_benchmark.pdf)