Sicherheitsmanagement für Webanwendungen
White Paper
Januar 2008

Rational. software

Sicherheitsrisiken für Ihre Webanwendungen

Highlights

Während die Abhängigkeit der Unternehmen von Webanwendungen ständig zunimmt, wird es gleichzeitig immer schwieriger, diese komplexen Komponenten adäquat zu schützen. Die meisten Unternehmen statten ihre Websites zwar mit Firewalls, Secure Sockets Layer (SSL) und Sicherheitsmechanismen für Netzwerke und Hosts aus, doch die Mehrheit der Angriffe zielt auf die Anwendungen selbst ab – und hier bieten die genannten Technologien keinen ausreichenden Schutz.

In dieser Veröffentlichung erfahren Sie, was Sie tun können, um Ihr Unternehmen besser zu schützen. Dabei wird eine Strategie zur Verbesserung der Webanwendungssicherheit in Ihrem Unternehmen beschrieben.

Was macht Webanwendungen anfällig?

Im OSI-Referenzmodell 1 bewegt sich jede Nachricht durch sieben Netzwerkprotokollschichten. Die Anwendungsschicht als oberste Ebene umfasst HTTP und andere Protokolle, die Nachrichten mit Inhalten übertragen, wie z. B. HTML, XML, Simple Object Access Protocol (SOAP) und Web-Services.

Diese Veröffentlichung konzentriert sich auf HTTP-basierte Anwendungsangriffe – eine Angriffstechnik also, gegen die traditionelle Firewalls keinen ausreichenden Schutz bieten. Viele Hacker verstehen es nämlich, HTTP-Anforderungen auf der Netzwerkebene harmlos aussehen zu lassen, während die enthaltenen Daten aber doch potenziell gefährlich sind. HTTP-basierte Angriffe können einen unbeschränkten Zugriff auf Datenbanken erwirken, Systembefehle aufrufen und sogar die Inhalte von Websites ändern.

Highlights

Ohne ein geregeltes Management von Sicherheitstests während des gesamten Anwendungsbereitstellungszyklus sind die Anwendungen eines Unternehmens permanent durch HTTP-basierte Angriffe gefährdet. Dies kann folgende Ursachen haben:

- *Analytiker und Architekten sehen das Thema Sicherheit in der Verantwortung der Teams für den Netzwerk- und IT-Betrieb, d. h., nur wenige Sicherheitsexperten im Unternehmen sind sich der Bedrohung auf Anwendungsebene bewusst.*
- *Teams formulieren die Anforderungen an die Anwendungssicherheit als vage Erwartungen oder Negativaussagen („Generell keine ungeschützten Eingangspunkte zulassen"), was die Einrichtung von Tests erschwert.*
- *Das Testen der Anwendungssicherheit erfolgt spät im Lebenszyklus – und dann nur für unbefugte Zugriffsversuche.*

Um Webanwendungen gegen Angriffe zu schützen, sollten Unternehmen allgemein vorbeugende Maßnahmen treffen und auch gezielt spezielle Technologien einsetzen.

Verbreitete Angriffe auf Webanwendungen

Die spezifischen Schwachstellen einer Webanwendung sollten die Technologie bestimmen, die zu ihrem Schutz eingesetzt wird. Abbildung 1 zeigt verschiedene Punkte im System, die geschützt werden müssen. Häufig ist es sinnvoll, zunächst allgemeine Gegenmaßnahmen zu treffen, um sicherzustellen, dass Sie sich für die Technologie entscheiden, die Ihren Anforderungen am besten gerecht wird, anstatt eine Lösung zu wählen, die den Anspruch erhebt, die neuesten Hacking-Verfahren bekämpfen zu können.

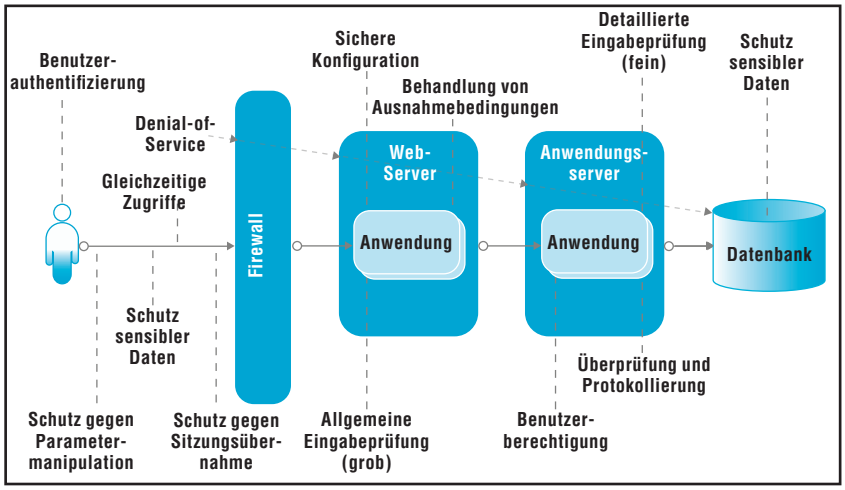

Abbildung 1: Aspekte der Webanwendungssicherheit.

Highlights

Unternehmen können verschiedene Abwehrmaßnahmen gegen Verstöße bei Webanwendungen treffen, die durch das Vortäuschen einer anderen Identität, die Manipulation von Daten während der Übertragung oder durch das Bestreiten von getätigten Transaktionen verursacht werden.

Tabelle 1 zeigt verbreitete Bedrohungen und entsprechende Abwehrmaßnahmen. Ihre eigenen Anwendungen können aber ganz anderen Risiken ausgesetzt sein.

Tabelle 1: Verbreitete Angriffe auf Webanwendungen

Beschreibung	Häufige Ursachen	Abwehrmaßnahmen
Vortäuschen einer anderen Identität (Impersonation)		
Eingabe der Berechtigungsnachweise eines anderen Benutzers oder Ändern eines Cookies oder Parameters, um die Identität dieses Benutzers anzunehmen oder um vorzugeben, das Cookie stamme von einem anderen Server	• Verwendung kommunikationsbasierter Authentifizierung, um auf die Daten beliebiger Benutzer zugreifen zu können • Verwendung nicht verschlüsselter Berechtigungsnachweise, die aufgezeichnet und wiederverwendet werden können • Speicherung von Berechtigungsnachweisen in Cookies oder Parametern • Verwendung unbestätigter Authentifizierungsverfahren oder Authentifizierung durch die falsche „Trust Domain“ • Verhinderung der Hostauthentifizierung durch die Client-Software	Einsatz strikter Authentifizierungs- und Schutzmechanismen für Berechtigungsinformationen durch: • Frameworks, die als Bestandteil von Betriebssystemen mitgeliefert werden • Verschlüsselte Tokens, wie z. B. Sitzungscookies • Digitale Signaturen
Manipulation von Daten während der Übertragung (Tampering)		
Ändern oder Löschen einer Ressource ohne entsprechende Berechtigung (z. B. Vandalisierung von Websites, Ändern von Daten während der Übertragung)	• Vertrauen gegenüber Datenquellen ohne vorherige Überprüfung • Bereinigung von Eingaben (Sanitizing), um die Ausführung von unerwünschtem Code zu verhindern • Ausführung mit erweiterten Berechtigungen (Privilege Escalation) • Unverschlüsselte Verarbeitung sensibler Daten	• Nutzung der Sicherheitsmechanismen des Betriebssystems zum Sperren von Dateien, Verzeichnissen und anderen Ressourcen • Überprüfung Ihrer Daten • Schutz der Daten während der Übertragung durch Hashverschlüsselung und Signaturen (z. B. durch SSL oder IPsec)
Bestreiten von getätigten Transaktionen (Repudiation)		
Versuch, Nachweise für eine erfolgte Aktivität zu vernichten, zu verbergen oder abzuändern (z. B. durch das Löschen von Protokollen oder durch das Anfordern von Änderungen unter einer falschen Identität)	• Verwendung eines unzureichenden Berechtigungs- oder Authentifizierungsprozesses (oder keines solchen Prozesses) • Unzureichende Protokollierung • Übertragung sensibler Informationen über nicht gesicherte Kommunikationskanäle	• Einsatz strikter Authentifizierung, Transaktionsprotokolle und digitaler Signaturen • Prüfungen

Highlights

Es können auch vorbeugende Maßnahmen gegen Angriffe getroffen werden, bei denen versucht wird, auf sensible Informationen zuzugreifen oder Serverressourcen zu überwinden.

Tabelle 1: Verbreitete Angriffe auf Webanwendungen

Beschreibung	Häufige Ursachen	Abwehrmaßnahmen
Weitergabe von Informationen (Information Disclosure)		
Weitergabe von personenbezogenen Daten (wie z. B. Kennwörtern oder Kreditkarteninformationen) sowie Informationen zur Anwendungsquelle und/oder ihren Hostmaschinen	• Bereitstellung eines authentifizierten Zugriffs auf die Daten anderer Benutzer • Übertragung sensibler Informationen über nicht gesicherte Kommunikationskanäle • Auswahl unzureichender Verschlüsselungsalgorithmen und -schlüssel	• Speicherung personenbezogener Daten auf Sitzungsebene (vorübergehend) anstatt auf permanenter Basis • Einsatz von Hashverfahren und Verschlüsselung für sensible Daten (wann immer möglich) • Abgleich von Benutzerdaten und Benutzerauthentifizierung
Serviceverweigerung (Denial-of-Service, DoS)		
• Überflutung – Versenden von so vielen Nachrichten oder Anforderungen gleichzeitig, dass dies eine Serverüberlastung verursacht • Sperrsituation (Lockout) – Erzeugen einer Anforderungsspitze, um die Serverantwortzeiten zu verlangsamen oder um einen Anwendungsneustart zu erzwingen	• Installation von zu vielen oder in Konflikt stehenden Anwendungen auf demselben Server • Vernachlässigung umfassender Einheitentests	• Filterung von Paketen durch eine Firewall • Einsatz von Lastausgleichsfunktionen zur Steuerung der Anzahl der Anforderungen einzelner Quellen • Verwendung asynchroner Protokolle für rechenintensive Anforderungen und Fehlerbehebung
Aneignung von Berechtigungen (Elevation of Privilege)		
Überschreiten der normalen Zugriffsberechtigungen, um Verwaltungsrechte oder den Zugriff auf Dateien mit vertraulichem Inhalt zu erlangen	• Ausführung von Web-Server-Prozessen als „Root“ oder „Administrator“ • Nutzung von Codefehlern, um Pufferüberläufe zu erzeugen oder um Anwendungen in einen Debugstatus zu versetzen	• Verfolgung einer Strategie, bei der grundsätzlich nur die erforderlichen Berechtigungen vergeben werden • Verwendung typsicherer Sprachen und Compileroptionen, um Pufferüberläufe zu verhindern oder zu steuern

Highlights

Durch die Anwendung verschiedener grundlegender Verfahren können Softwareentwicklungsteams dazu beitragen, verbreitete Sicherheitsverstöße bei Webanwendungen zu vermeiden und die Kosten für Fehlerkorrekturen zu senken.

Allgemeine Richtlinien zur Erzielung sicherer Webanwendungen

Durch den Einsatz von sicherheitsspezifischen Prozessen zur Erstellung von Anwendungen können Softwareentwicklungsteams zur Vermeidung von Sicherheitsverstößen beitragen, wie sie in Tabelle 1 aufgeführt sind. Konkret können Sie verschiedene allgemeine Richtlinien auf bestehende, neue oder geänderte Anwendungen im gesamten Prozess anwenden, um höheren Schutz zu erzielen und die Kosten der Fehlerkorrektur zu senken. Beispiele:

- *Ermitteln und Erarbeiten Sie Referenzwerte: Erfassen Sie den Bestand an Anwendungen und Systemen vollständig, einschließlich technischer Informationen (z. B. Internet Protocol (IP), Domain Name System (DNS), verwendetes Betriebssystem) und geschäftsbezogener Informationen („Wer genehmigte die Implementierung?“, „Wer sollte benachrichtigt werden, wenn die Anwendung ausfällt?“). Als Nächstes müssen Sie die Webinfrastruktur auf gängige Sicherheitslücken und deren Ausnutzung durch Angreifer überprüfen. Informieren Sie sich mit Hilfe Schwarzer Listen und Bug-Tracking-Sites über bekannte Angriffe auf Ihr Betriebssystem, Ihren Web-Server oder Produkte anderer Anbieter. Bevor Sie Ihre Anwendung auf einen Server laden, stellen Sie sicher, dass der Server über die erforderlichen Patches und Sicherheitseinrichtungen verfügt und per Scanning überprüft wurde. Scannen Sie anschließend Ihre Anwendung auf Anfälligkeiten für bekannte Angriffe, und berücksichtigen Sie dabei besonders HTTP-Anforderungen und andere potenzielle Verfahren zur Datenmanipulation. Testen Sie abschließend die Funktionen für die Anwendungsauthentifizierung und das Management von Benutzerrechten, und beenden Sie unbekannte Services.*
- *Bewerten Sie Risiken, und ordnen Sie sie zu: Stufen Sie Anwendungen und Systeme nach ihrem Risiko ein – mit besonderer Berücksichtigung der Datenspeicher, Zugriffssteuerung, Benutzereinrichtung und des Berechtigungsmanagements. Priorisieren Sie die bei der Bewertung ermittelten Sicherheitslücken der Anwendungen. Überprüfen Sie die Einhaltung von unternehmens- oder branchenspezifischen Vorgaben und gesetzlichen Vorschriften. Dadurch lassen sich akzeptable und nicht akzeptable Abläufe ermitteln.*
- *Schirmen Sie Ihre Anwendungen ab, und vermeiden oder begrenzen Sie Schäden: Bleiben Sie ständig auf dem Laufenden, was aktuelle Sicherheitsbedrohungen angeht, und wenden Sie alle verfügbaren Patches auf Ihre Anwendungen und/oder die Infrastruktur an. Wenn Sie ein Sicherheitsproblem nicht korrigieren können, verwenden Sie eine Anwendungsfirewall, schränken Sie den Zugriff ein, inaktivieren Sie die Anwendung, oder verlagern Sie sie, um das Sicherheitsrisiko zu minimieren.*
- *Überwachen und prüfen Sie Ihre Umgebung ständig: Planen Sie Bewertungen (Assessments) als festen Bestandteil Ihres dokumentierten Change-Management-Prozesses ein. Wenn Sie eine Erkennungsphase abgeschlossen haben, leiten Sie unmittelbar die nächste ein.*

Highlights

Der Rational Unified Process stellt ein umfassendes iteratives Framework für die Entwicklung von Webanwendungen auf der Basis bewährter Verfahren der Branche zur Verfügung.

Der Lebenszyklus von Webanwendungen

Abbildung 2 zeigt den IBM Rational Unified Process (IBM RUP). Die Lösung stellt ein vielfach eingesetztes iteratives Prozessframework für die Webanwendungsentwicklung auf der Basis bewährter Verfahren der Branche zur Verfügung. Das Framework – das möglicherweise zwei oder mehr Iterationen erfordert, setzt sich aus folgenden Phasen zusammen:

- *Konzeption (Inception): Durchführung einer Kosten-Nutzen-Analyse (Business Case), Definition des Umfangs (Scope) und eines strategischen Konzepts (Operational Vision). Daran schließt sich die Erstellung eines ersten Anwendungsfallmodells, eines Projektplans sowie einer Risikobewertung und Projektbeschreibung an, einschließlich Basisanforderungen, Sicherheitsanforderungen (z. B. Klärung von Sicherheitsstrategien und der Vorgehensweise zur Einhaltung von Sicherheitsbestimmungen), Einschränkungen, Features und mögliche Architekturprototypen.*
- *Ausarbeitung (Elaboration): Konkretisierung der Vision, Berücksichtigung architekturrelevanter Szenarios und Detaillierung des Anwendungsfall modells. Darauf folgt das Erstellen und Testes eines (oder mehrerer) Prototypen zur Minderung technischer Risiken.*
- *Konstruktion (Construction): Entwicklung detaillierter Designs für spezifische Komponenten und ihre Interaktionen mit anderen Anwendungen, wobei ein permanenter Abgleich mit den Anforderungen erfolgt. Generierung von Code und Testkomponenten zur Erzielung hoher Leistung, Zuverlässigkeit und Sicherheit – mit gleichzeitiger Problemverfolgung und -behebung – und Integration der getesteten Komponenten in ein erstes Release.*
- *Übergang (Transition): Verteilung der Anwendung, Schulung der Benutzer und Durchführung von Betatests, um Sicherheit und Leistung sicherzustellen und um zu prüfen, ob die Anwendung die Anforderungen erfüllt. Wenn dann kontinuierlich Änderungen an der Anwendung vorgenommen werden, müssen die Leistung, Zuverlässigkeit und Sicherheit überwacht werden.*

Highlights

Jede der vier Phasen des Rational Unified Process – Konzeption, Ausarbeitung, Konstruktion und Übergang – beinhaltet selbst wieder mehrere Disziplinen und erfordert möglicherweise mehrere Iterationen.

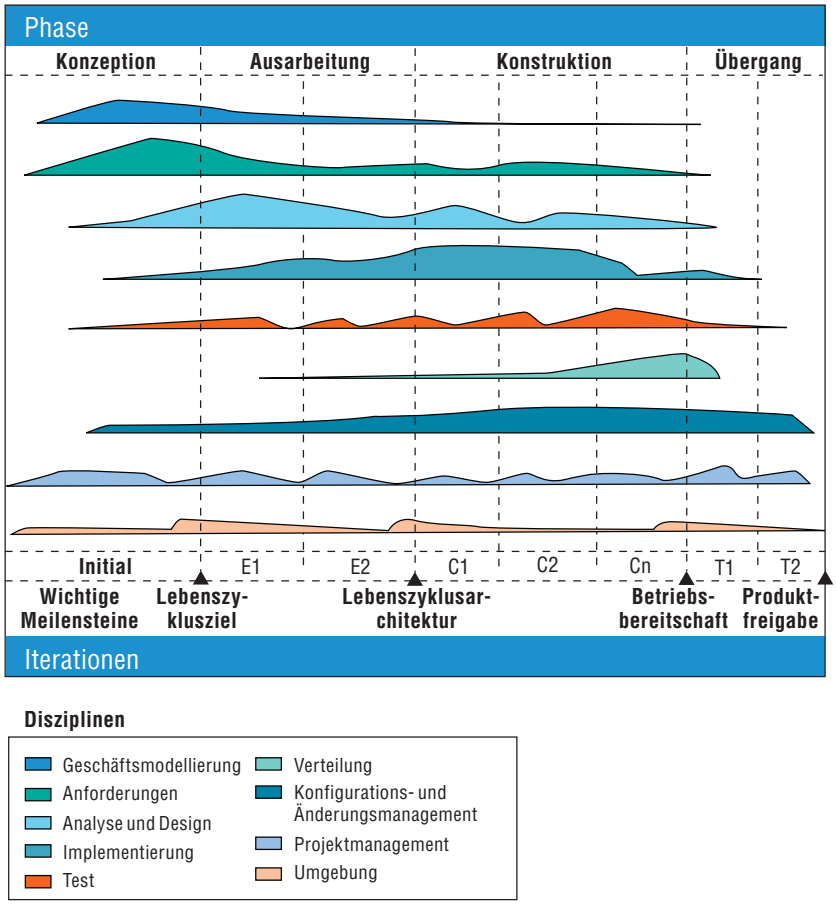

Abbildung 2: Phasen, Disziplinen und Meilensteine im IBM Rational Unified Process.

Highlights

Unternehmen können das RUP-Framework über alle Phasen des Webanwendungslebenszyklus hinweg verwenden, um Sicherheitslücken proaktiv zu erkennen und zu korrigieren, bevor die sich ergebenden Probleme sehr komplex werden und dadurch hohe Kosten verursachen.

Wenn der allgemeine Wettbewerbsdruck dazu führt, dass Unternehmen ihre Webanwendungen ohne adäquate Sicherheitstests entwickeln, können gefährliche Sicherheitslücken unbeobachtet bleiben – mit den entsprechenden Risiken für die Unternehmen.

Sicherheitstests im Anwendungslebenszyklus

Durch Anwendung der Richtlinien des RUP-Frameworks in einem frühen Stadium des Webanwendungslebenszyklus, können Sie Sicherheitslücken dann ermitteln und beheben, wenn es noch am kostengünstigsten ist. Später im Entwicklungs- und Bereitstellungsprozess werden Fehler dann schnell komplexer, und die Fehlerbehebung wird teurer.

Beispielsweise kann es sein, dass die Teams, die die Anforderungen während der Konzeptions- und Ausarbeitungsphase sammeln, nicht die gängigen Sicherheitsbedrohungen für Webanwendungen kennen und deshalb auch nicht darauf achten, die entsprechenden Anforderungen zur Begegnung dieser Risiken zu definieren. Wenn keine konkreten Sicherheitsanforderungen auf Anwendungsebene definiert wurden, kann es in der Konstruktionsphase dazu kommen, dass die Programmierer fehlerhaften Code wiederverwenden oder beim Schreiben von neuem Code einen IDE-Assistenten verwenden, der Sicherheitsaspekte nur unzureichend berücksichtigt. Mit der Folge, dass die Daten nicht genau genug überprüft oder Sicherheitsfunktionen im Anwendungsframework nicht richtig interpretiert werden. In der Übergangsphase vertrauen Unternehmen die Überprüfung von Webanwendungen häufig einer kleinen Zahl von Sicherheitsexperten an, die versuchen, Sicherheitslücken noch vor der Verteilung der Anwendungen festzustellen, was zu Prozessengpässen führt. Darüber hinaus ist die Behebung von Fehlern, die erst in diesem Stadium erkannt werden, in vielen Fällen mit einem hohen Zeit- und Kostenaufwand verbunden.

Wie die Zahlen in Tabelle 2 2 zeigen, kostet ein Designfehler, der erst nach Verfügbarkeit einer Webanwendung festgestellt wird, etwa 30 mal mehr – verglichen mit den Kosten, die entstehen, wenn der Fehler bereits während des Designs behoben wird. Und bei diesen Schätzungen sind noch nicht die Kosten eingerechnet, die beispielsweise durch verlorene Marktanteile, Imageschäden oder unzufriedene Kunden zusätzlich entstehen.

Highlights

Die Korrektur eines Designfehlers nach der Implementierung einer Webanwendung kostet etwa 30 mal mehr – verglichen mit den Kosten, die entstehen, wenn der Fehler bereits während des Designs behoben wird.

Um hohe Kosten für die Fehlerkorrektur zu vermeiden, können Unternehmen Sicherheitstests für Anwendungen in ihren Entwicklungs- und Bereitstellungsprozess integrieren.

Tabelle 2: Relative Kosten für die Fehlerkorrektur abhängig vom Zeitpunkt der Erkennung

Art des Fehlers	Design	Codierung	Integration	Beta	Verteilung
Design	1x	5x	10x	15x	30x
Codierung		1x	10x	20x	30x
Integration			1x	10x	20x

Wahl des am besten geeigneten Testverfahrens

Um hohe Kosten für die Fehlerkorrektur zu vermeiden, müssen Unternehmen verschiedene Anwendungssicherheitstests (wie in Tabelle 3 gezeigt) zusammen mit anderen QM-Maßnahmen in ihren Entwicklungs- und Bereitstellungsprozess integrieren.

Tabelle 3: Sicherheitstestverfahren für Webanwendungen

Beschreibung	Vorteile	Nachteile
Manuelles Verfahren		
Penetration- und Sicherheitsabnahmetests durch eine kleine Gruppe von Sicherheitsexperten, die bekannte Tools und Scripts einsetzen	Erstellung gezielter Tests für spezifische Anwendungsfunktionen	• Einschränkung der Testaktivitäten auf Fachleute, was zu Engpässen führen kann • Fehleranfälliger Prozess und evtl. neu entstehenden Kosten • Zeitliche Beschränkung, die zur Folge hat, dass nicht alle Anwendungsbereiche abgedeckt werden können
Automatisierte Verfahren		
Normalerweise nach einem der folgenden Prinzipien: • Bottom-up – Spezifische Tests für einzelne Funktionen, erstellt vom Codeentwickler • Top-down – Tests, die von QA-Teams aus der Perspektive der Endbenutzer erstellt werden	Aufwandsentlastung mit Qualitätsverbesserungen und reduziertem Aufwand für Abnahmetests und iterativen Entwicklungsprozessen	Höherer Aufwand als bei der Einrichtung und Durchführung manueller Tests

Highlights

Blackbox- und Whitebox-Testverfahren können handelsübliche Tools nutzen, während GrayboxTestverfahren ein maßgeschneidertes Anwendungsframework voraussetzen.

Beschreibung	Vorteile	Nachteile
Blackbox oder System		
Ausschließliche Berücksichtigung von Systemeingabe und -ausgabe durch Modifizieren der normalen Benutzereingabe, damit sich die Anwendung unerwartet verhält	Nutzung etablierter automatisierter Testtools, für deren Verwendung nur minimales Anwendungswissen erforderlich ist	• Einsatz nur möglich, wenn alle Anwendungskomponenten für den Test bereitstehen (spät in der Staging-Phase oder in der Produktionsumgebung) • Evtl. Generierung von Transaktionen, die durch geänderte Benutzereingabe schwer zu ignorieren oder zurückzunehmen sind • Evtl. Verschleierung von Fehlern durch begrenzten Einblick in die Anwendung
Whitebox oder Quelle		
Bewertung einzelner Komponenten hinsichtlich bestimmter Funktionsfehler, häufig in Kombination mit Code-Scanning-Tools und Peer-Reviews	Einsatz von Tools mit bewährter Integration in Entwicklungsumgebungen – mit der Möglichkeit zur effektiven Fehlererkennung bei den getesteten Funktionen	• Keine Erkennung von Designfehlern oder nicht erfüllten Anforderungen • Evtl. keine Berücksichtigung von Sicherheitslücken bei Angriffen, die mehrere Komponenten oder bestimmte zeitliche Abläufe betreffen, die nicht von Einheitentests abgedeckt werden • Annahme, dass sich Programmierer der Notwendigkeit von Sicherheitstests bewusst sind
Graybox (Verwendung eines von der Anwendung definierten Frameworks)		
Kombination von Blackbox- und Whitebox-Verfahren zur Ausführung von Tests, die nicht über handelsübliche Tools erhältlich sind	• Bereitstellung eines sehr umfassenden Verfahrens durch die Kombination von System- und Einheitentests • Bereitstellung von zustands und zeitbasierten Tests und Verwendung von Agenten und Proxys • Integration des Frameworks in die Anwendung, um den Datenfluss zu überwachen und Prüfungen durchzuführen, ohne die Produktionsdaten zu beeinträchtigen	• Notwendigkeit eines Frameworks, das in der Konzeptionsphase und während der Designaktivitäten definiert werden muss • Evtl. gleicher Aufwand für den Aufbau des Testframeworks wie für die Anwendungsentwicklung

Highlights

Externe Berater können Unternehmen bei den entsprechenden Schulungs-, Kommunikations- und Überwachungsaktivitäten unterstützen, um das Sicherheitsbewusstsein der Mitarbeiter zu schärfen.

Vier bewährte Verfahren für den strategischen Schutz von Webanwendungen

Wenn es um die Sicherheit von Webanwendungen geht, können sich Unternehmen im Prinzip für eine von vier bewährten Strategien entscheiden. Diese sind im Folgenden beschrieben:

1. Schärfung des Sicherheitsbewusstseins

Dies umfasst Schulungs-, Kommunikations- und Überwachungsaktivitäten – vorzugsweise in Zusammenarbeit mit einem Berater.

Schulung

Dieser Bereich umfasst folgende Aktivitäten: Angebot an jährlich wiederkehrenden Sicherheitsschulungen für alle Mitarbeiter im Anwendungsteam: Entwickler, QA-Spezialisten, Analytiker und Manager; Beschreibung der aktuellen Angriffe und eines empfohlenen Prozesses zur Fehlerkorrektur; Diskussion der aktuellen Sicherheitsverfahren des Unternehmens; Erwartung an die Entwickler, an den entsprechenden Schulungen teilzunehmen, damit sie die vordefinierten Sicherheitsfunktionen des Frameworks effektiv nutzen können; Benutzerschulung zu handelsüblichen Sicherheitstools anhand des Schulungsmaterials der entsprechenden Anbieter sowie Einbindung von Schulungen zum Thema Sicherheit in den Projektplan.

Highlights

Kommunikation:
Dieser Bereich umfasst folgende Aktivitäten: Erfassung bewährter Sicherheitsverfahren in allen Teams und Geschäftsbereichen Ihres Unternehmens; Verteilung dieser Informationen in einem kurzen Dokument und komfortable Bereitstellung im Intranet; frühe Einbeziehung der IT-Sicherheitsexperten und Entwicklung von Prozessen, die ein Fachleute-Mentoring (Peer Mentoring) einschließen; Herstellung von Beziehungen zwischen dem Sicherheitsteam und den einzelnen Anwendungsteams zur Unterstützung bei Fragen zu den Anforderungen und zum Design von Anwendungen.

Überwachung:
Dieser Bereich umfasst folgende Aktivitäten: Permanentes Informieren der Manager über den Sicherheitsstatus jeder einzelnen Anwendung in der Produktion; Verfolgung von Sicherheitsproblemen durch Ihre bestehende Infrastruktur zur Fehlererfassung und -meldung, um für alle Beteiligten die erforderliche Transparenz zu schaffen.

Bei der Zuordnung ihrer begrenzten Sicherheitsressourcen können Unternehmen nach Risiken und Zuständigkeiten priorisieren.

2. Kategorisierung der Anwendungsrisiken und -zuständigkeiten
Jedes Unternehmen verfügt nur über begrenzte Ressourcen und muss deshalb Prioritäten setzen. Um Prioritäten im Bereich der Sicherheitsinfrastruktur zu setzen, gibt es folgende Möglichkeiten:

- *Definition von Risikoschwellen und Festlegung des Zeitpunkts, zu dem das Sicherheitsteam die Anwendungsservices terminiert*
- *Kategorisierung der Anwendungen nach Risikofaktoren (z. B. Internet oder Intranet im Vergleich zu einem Extranet)*
- *Regelmäßige Erstellung von Risikoberichten auf der Basis von Sicherheitsscans, die Problemstellungen mit definierten Risikoschwellen abgleichen*
- *Pflege einer Datenbank, mit der Anwendungen nach Risiko analysiert und eingestuft werden können – d. h., die Teams können darüber informiert werden, wie ihre Anwendungen gegenüber den implementierten Systemen abschneiden*

Highlights

Zur Steuerung der Entwicklungs- und Bereitstellungsprozesse und für das Compliance-Management müssen Unternehmen ein Sicherheitsprogramm einrichten und eine „Null-Toleranz-Strategie" definieren.

Durch die Integration von Sicherheitstests im gesamten Softwarebereitstellungszyklus können Unternehmen das Design und die Entwicklung von Anwendungen sowie die zugehörigen Anwendungstests optimieren.

3. Konsequente Strategieumsetzung (Zero Tolerance)

Als grundlegendes Instrument zur Steuerung des Entwicklungs- und Bereitstellungsprozesses kann eine klar definierte Sicherheitsstrategie zur Verringerung der Risiken beitragen, die mit der Implementierung anfälliger oder nicht konformer Anwendungen verbunden ist. Während der Konzeptionsphase ist festzulegen, welche Tests die Anwendung vor ihrer Verteilung bestehen muss, und alle Teammitglieder sind zu informieren. In der Konzeptions- und Ausarbeitungsphase müssen die Anforderungs- und Designspezifikationen für Sicherheitsproblemstellungen formal geprüft werden, bevor der Anwendungscode geschrieben wird. Sicherheitsausnahmen sind nur während des Designs und nur mit Genehmigung der entsprechenden Führungskräfte zulässig.

4. Integration von Sicherheitstests über den gesamten Entwicklungs- und Bereitstellungsprozess hinweg

Die Integration von Sicherheitstests in den gesamten Bereitstellungszyklus bietet viele Vorteile für das Design und die Entwicklung von Anwendungen sowie für die zugehörigen Anwendungstests. Sie sollten Ihre funktionalen Anforderungen von Sicherheitstests abhängig machen, die Ihre Anwendung bestehen muss, um sicherzustellen, dass Ihr Framework folgende Voraussetzungen erfüllt:

- *Nutzung automatisierter Tools und Möglichkeit zur Ausführung zu jedem beliebigen Zeitpunkt im Entwicklungs- und Bereitstellungsprozess.*
- *Möglichkeit zur Ausführung von Einheiten- und Systemtests sowie von Tests auf Anwendungsebene.*
- *Möglichkeit zu Tests (Audit Testing) während des Produktionsbetriebs.*
- *Möglichkeit zur Durchführung ereignisgesteuerter Tests.*
- *Verwendung einer dynamischen Entwicklungsmethodik für Sicherheitsprozeduren.*
- *Möglichkeit zur Ausführung während der Programmierung, bei Tests und zeitgleich mit Integrations- und Produktionsaktivitäten.*

Highlights

Während der Konzeptionsphase können Unternehmen Anforderungen strukturieren, die sich an verschiedene Sicherheitsprobleme auf Anwendungsebene richten.

Tabelle 4 zeigt verschiedene Möglichkeiten zur Anforderungsstrukturierung, mit denen während der Konzeptionsphase verschiedene Sicherheitsprobleme auf Anwendungsebene adressiert werden können.

Tabelle 4: Konzeption – Definition der Sicherheitsanforderungen

Problem auf Anwendungsebene	Anforderungen und Einschränkungen
Anwendungsumgebung	• Ermittlung, Klärung und Umsetzung der Sicherheitsstrategien Ihres Unternehmens • Erkennung von Infrastruktureinschränkungen (z. B. Services, Protokolle und Firewalls) • Ermittlung von Einschränkungen der Hosting-Umgebung (z. B. Virtual Private Network (VPN), Sandboxing) • Definition der Konfiguration der Anwendungsverteilung • Definition der Netzwerkdomänenstrukturen, des Clustering und ferner Anwendungsserver • Ermittlung von Datenbankservern • Ermittlung sicherer Kommunikationsfeatures, die von der Umgebung unterstützt werden • Hinweise zu Web-Server-Farmen (z. B. Management des Sitzungsstatus, maschinenspezifische Chiffrierschlüssel, SSL, Fragen der Zertifikatsimplementierung und Roaming-Profile); wenn die Anwendung SSL verwendet, müssen die Zertifizierungsstelle (CA) und die zu verwendenden Typen angegeben werden. • Berücksichtigung von Fragen der Skalierbarkeit und Leistung • Untersuchung des Code-Trust-Level
Eingabe-/ Datenprüfung und -authentifizierung	• Grundsätzliche Einstufung der gesamten Clienteingabe als potenziell gefährlich • Ermittlung aller „Trust Boundaries“ für Identitätsaccounts und/oder Ressourcen, die diese Grenzen überschreiten • Definition von Account-Management-Strategien und einer „Least-privileged Accounts“-Strategie • Angabe von Anforderungen für sichere Kennwörter und Umsetzungsmaßnahmen • Verschlüsselung von Berechtigungsnachweisen mit SSL, VPN, IPsec usw. und Verhindern der Übertragung von Authentifizierungsinformationen (wie z. B. Tokens, Cookies und Tickets) über nicht verschlüsselte Verbindungen • Rückgabe minimaler Fehlerinformationen an den Client, wenn ein Authentifizierungsfehler auftritt
Session-Management	• Begrenzung der Sitzungsdauer • Schutz des Sitzungsstatus gegen unbefugten Zugriff • Verhinderung der Übertragung von Sitzungs-IDs in Abfragezeichenfolgen

Highlights

Maßnahmen zur Prüfung von Codierung und Daten bieten erhebliche Vorteile für die Ausarbeitungs- und Konstruktionsphase des Prozesses der Softwareentwicklung und -bereitstellung.

Tabelle 5 beschreibt Aktivitäten während der Ausarbeitungs- und Konstruktionsphase, die definierten Sicherheitsanforderung entsprechen.

Tabelle 5: Ausarbeitung und Konstruktion – Modellierung und Codierung von Sicherheitsmechanismen

Sicherheitsaspekt	Ausarbeitung	Konstruktion
Programmierverfahren		• Kein Reduzieren oder Ändern der Standardsicherheitseinstellungen ohne vorherige Prüfung der möglichen Auswirkungen • Kein Hinterlassen geheimer Informationen im Code (in der Hoffnung, dass sie unbeachtet bleiben) • Kein Offenlegen von Informationen, die nicht benötigt werden • Häufiges Testen auf Sicherheitsprobleme und frühestmögliche Problembehebung • Behandlung von Fehlern möglichst in einem sicheren Modus; es sollten weder Stack-Traces angezeigt werden, noch sollten sensible Daten ungeschützt sein
Eingabe-/ Datenprüfung	• Einstufung der gesamten Clienteingabe als verdächtig; Überprüfung der Eingabe auf einem Server, der von der Anwendung gesteuert wird – selbst dann, wenn auch auf der Clientseite geprüft wird • Berücksichtigung potenzieller Probleme durch SQL-Injektion und Cross-site Scripting • Ermittlung von Eingangspunkten und „Trust Boundaries“	• Überprüfung aller Eingabeparameter, z. B. Formularfelder, Abfragezeichenfolgen, Cookies und HTTP-Header • Verfolgung einer Strategie, bei der nur bekannterweise einwandfreie Eingabe akzeptiert und bekannterweise verdächtige Eingabe zurückgewiesen wird • Überprüfung der Daten nach Typ, Länge, Format und Bereich • Sicherstellen, dass Ausgaben, die Benutzereingaben beinhalten, stets sauber HTML- oder URL-codiert sind

Highlights

Durch verbesserte Authentifizierungs-, Berechtigungs- und Konfigurationsmanagementverfahren können sich Unternehmen bereits in der Ausarbeitungs- und Konstruktionsphase um Sicherheitsprobleme kümmern.

Sicherheitsaspekt	Ausarbeitung	Konstruktion
Authentifizierung	• Trennung des Zugriffs auf öffentliche und eingeschränkt zugängliche Bereiche, ID-Accounts und Ressourcen, die „Trust Boundaries“ überschreiten • Ermittlung von Accounts, die die Anwendung unterstützen oder verwalten • Verschlüsselung und sichere Speicherung der Berechtigungsnachweise von Benutzern • Angabe der ID zur Authentifizierung bei der Datenbank	• Speicherung von Kennwörtern als „Digests“ • Rückgabe minimaler Fehlerinformationen bei Authentifizierungsfehlern • Keine Verwendung von HTTP-Headerinformationen aus Sicherheitsgründen
Berechtigung	• Angabe aller IDs und Ressourcen, auf die alle zugreifen können • Ermittlung berechtigter Ressourcen und berechtigter Operationen • Trennung der Berechtigungen für verschiedene Rollen (z. B. Integration von Berechtigungsgranularität) • Ermittlung von Sicherheitsanforderungen für den Codezugriff	• Beschränkung der Datenbankanmeldung auf zugriffsspezifische gespeicherte Prozeduren; kein direkter Zugriff auf Tabellen • Beschränkung des Zugriffs auf Ressourcen der Systemebene
Konfigurationsmanagement	• Schutz der Verwaltungsschnittstellen und der Kanäle für die Fernverwaltung durch strikte Authentifizierungs- und Berechtigungsfunktionen • Bereitstellung rollenbasierter Administratorberechtigungen • Anwendung des „Least-privileged“ -Prozesses und von Service-Accounts	• Sicherung von Konfigurationsspeichern • Kein Speichern vertraulicher Daten in unverschlüsselten Konfigurationsdateien

Highlights

Auch Sitzungsschutz, Ausnahmemanagement sowie Prüfungs- und Protokollierungsfunktionen tragen zu einer höheren Sicherheit von Webanwendungen bei.

Sicherheitsaspekt	Ausarbeitung	Konstruktion
Schutz sensibler Daten und Sitzungen	• Vermeidung des Speicherns geheimer Daten; Ermittlung von Verschlüsselungsalgorithmen und Schlüsselgrößen für alle Daten, die aufbewahrt werden müssen • Ermittlung von Schutzmechanismen für sensible Daten, die über das Netzwerk übertragen werden • Einsatz von SSL zum Schutz von Authentifizierungscookies und zur Verschlüsselung ihres Inhalts • Ermittlung einer Methodik zum Schutz von Chiffrierschlüsseln und zur Verwendung von ausschließlich bekannten Verschlüsselungsbibliotheken und -services • Ermittlung der richtigen Verschlüsselungsalgorithmen und Schlüsselgröße	• Kein Speichern sensibler Daten im Code • Kein Speichern von Datenbankverbindungen, Kennwörtern oder Schlüsseln in unverschlüsselter Form • Kein Protokollieren sensibler Daten in unverschlüsselter Form • Kein Speichern sensibler Daten in Cookies oder Übertragen solcher Daten in Form einer Abfragezeichenfolge oder eines Formularfelds
Ausnahmemanagement	• Definition einer Standardstrategie für die strukturierte Behandlung von Ausnahmebedingungen • Angabe generischer Fehlernachrichten, die an den Client zurückgegeben werden	• Offenlegung von möglichst wenigen Informationen nach Auftreten einer Ausnahmebedingung
Überprüfung und Protokollierung	• Ermittlung von Schlüsselparametern für die Überprüfung und Protokollierung • Angabe von Speicher-, Sicherheits- und Analysefeatures für Protokolldateien • Festlegung, in welcher Art und Weise die Aufrufidentitäten über die verschiedenen Ebenen (Tiers) weitergeben werden – auf Betriebssystem- oder Anwendungsebene	• Keine Protokollierung sensibler Daten

Highlights

Durch die Möglichkeit zur Durchführung ereignisgesteuerter Tests können Unternehmen die erforderlichen Sicherheitstestfunktionen bei der Entwicklung direkt in die Anwendungen integrieren.

Sie haben nicht nur die Möglichkeit, Sicherheit zu einem integralen Bestandteil der Anwendungsentwicklung und -bereitstellung zu machen. Sie können die erforderlichen Sicherheitstests bei der Entwicklung direkt in die Anwendung integrieren und so ereignisgesteuerte Tests durchführen. D. h., wenn hier ein Benutzer eine Anforderung stellt und die Anwendung antwortet, vergleicht die Testfunktion die Antwort mit einer erwarteten oder zuvor gespeicherten Antwort, um festzustellen, ob das System ordnungsgemäß arbeitet. In Abbildung 3 beispielsweise verwendet eine Anwendung eine Datenbank als Back-End-Komponente. Der Tester integriert einen „Spy Proxy“ und eine Prüffunktion in den Anforderungsablauf, und die Prüffunktion erhält Informationen darüber, wie eine normale Anforderung aussehen sollte. Die Prüffunktion kann dann die tatsächlichen Anforderungen des „Spy Proxy“ mit diesen Informationen vergleichen.

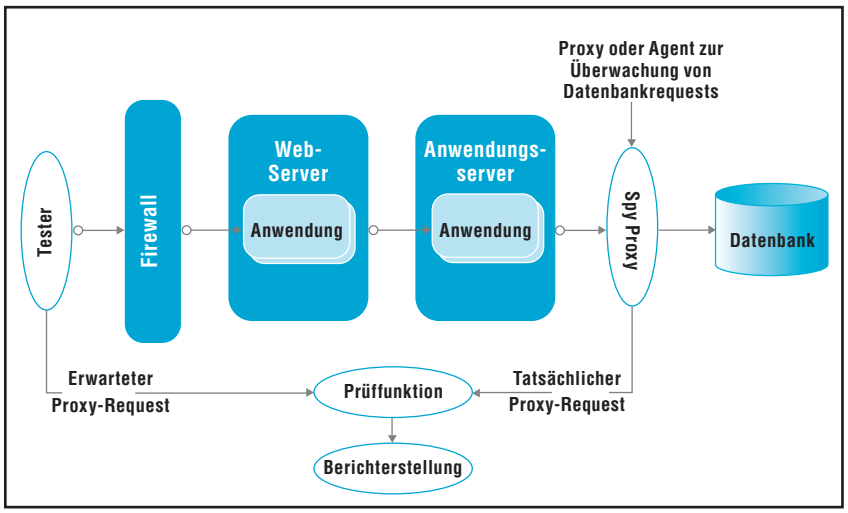

Abbildung 1: Aspekte der Webanwendungssicherheit.

Jeder Service (E-Mail-, XML- oder traditioneller Service) kann als Back-End dienen. Wie Sie den Code zur Prüfung der Anforderungen implementieren, hängt von der Anwendungsarchitektur ab. Beispielsweise könnte Ihre „Spy“ -Komponente ein Pseudodatenzugriffsobjekt, ein Proxy oder eine Klasse sein, die aus dem Front-End-Service abgeleitet wird. Sie können auch speziell für einen Test Code erstellen, den Sie in den Datenstrom integrieren, um Berichtsdaten für das Testframework zur Verfügung zu stellen. Durch die Koordination der Testobjekte erhalten Sie eine umfassende und differenzierte Kontrolle über unterschiedliche Tests. Sie können die Tests unter Verwendung von Blackbox- oder Whitebox-Verfahren durchführen. Dadurch erhöht sich die Wahrscheinlichkeit, dass Sicherheitsprobleme schon in einem frühen Stadium erkannt werden – also bevor sie ein ernstzunehmendes Risiko für Ihr Unternehmen darstellen.

Weitere Informationen

Weitere Informationen zum IBM Rational-Konzept und dazu, wie Sie sichere Webanwendungen mit den automatisierten IBM Rational-Lifecycle-Sicherheitstools erstellen können, erhalten Sie von Ihrem IBM Ansprechpartner oder auf folgender Website:

ibm.com/software/rational/offerings/testing/webapplicationsecurity

IBM Deutschland GmbH
70548 Stuttgart
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

[1] International Organization for Standardization; www.iso.org
[2] www.nist.gov/director/prog-ofc/report02-3.pdf

Gedruckt in den USA
01-08

RAW14010-DEDE-00