GAI NetConsult GmbH— Am Borsigturm 58 — 13507 Berlin — www.gai-netconsult.de — Tel: 030/417898-0

Sicherheit von Web-Applikationen – Therapieren von Symptomen oder Schaffung einer gesunden (sicheren) Basis?

Wilfrid Kettler (GAI NetConsult GmbH)

August 2009 Sonderdruck aus Security *Journal* #44

Die Zahl der Web-Applikationen im Enterprise Business wächst immens, auch die Vernetzung und Verflechtung von Geschäftsprozessen wird stetig enger, ohne dabei die Komplexität zu reduzieren, im Gegenteil. Entwickler werden mehr denn je mit der Realisierung neuer Anwendungen ausgelastet sein und sich weniger um die „Reparatur" von Sicherheitslücken in vorhandenen Systemen kümmern können – diese Situation dürfte die Anbieter von externen Schutzsystemen, wie WAFs (Web Application Firewalls) und Dienstleister im Umfeld von „virtual Patching" mit viel Arbeit versehen. Der nachfolgende Artikel beleuchtet diese Problematik aus Sicht des Software-Entwicklers.

Mit der zunehmenden Zahl neu entstehender Web-Applikationen wird aller Wahrscheinlichkeit nach auch die entsprechende Zunahme von Sicherheitsverletzungen in und mit diesen Systemen wachsen – es sei denn, die Softwareentwickler nehmen die Herausforderung an, sichere(re) Software zu erstellen und stoppen diesen Trend bzw. können ihn eventuell sogar umkehren.
Ist sichere Software heute überhaupt machbar? Kann man diese Aufgabe meistern, trotz permanent neu identifizierter Schwachstellen und immer größerer Komplexität der Anwendungen? Oder muss man mit der (vermeintlichen) Erkenntnis weiter leben, dass es ohnehin nie fehlerfreie und auch nie sichere Software geben wird?

These: Fehlerhafte Software ist selten sicher

Die meisten Sicherheitslücken resultieren aus Fehlern, die unbeabsichtigt während des Entwurfs und der Entwicklung der Software gemacht wurden. Um die Verwundbarkeit von Software grundsätzlich zu verringern, muss daher die gesamte Fehlerrate in der Software verringert werden. Das SEI (Software Engineering Institute der Carnegie Mallon University) hat mehrere tausend Programme auf Fehlerraten analysiert. Im Ergebnis wurde pro 7 bis 10 Lines of Code durchschnittlich 1 Fehler gefunden. Selbst wenn 99% dieser Fehler wieder entfernt werden würden, bevor die Software freigegeben wird, verblieben dennoch statistisch ca. 1 bis 1,5 Fehler pro 1.000 Zeilen des neuen oder geänderten Codes.
Tatsächlich verbleiben jedoch – nach Auswertung von ebenfalls mehreren hundert Softwareprojekten durchschnittlich 1 bis 7 Fehler pro 1.000 Programmzeilen in der freigegebenen Software bestehen [Quelle: Jones]. Folgt man der Analyse der CERT® Gruppe des SEI, dann werden über 90% von Sicherheitslücken in Softwaresystemen durch lang bekannte **Programmierfehler** verursacht – selbstverständlich gibt es zusätzlich auch neue, bislang nicht als problematisch bewertete Ursachen; diese sind letztlich noch hinzuzurechnen. Die Studie belegt, dass allein die „Top10" der längst bekannten Ursachen ca. 75% aller Vulnerabilities ausmachen.
Eine andere Analyse von 45 eBusiness Anwendungen ergab, dass 70% der Sicherheitsverletzungen auf Entwurfsfehler der Software zurückzuführen waren [Quelle: Jacquith].
Einige Probleme sind auch auf die fehlerhafte Nutzung anspruchsvoller, komplexerer Architektur- und **Entwurfskonstrukte**, zurückzuführen. Dies führt zu:

- unzulänglicher Authentisierung
- und Autorisierung,
- falscher Anwendung von Ver-
- schlüsselungsverfahren,
- ungenügendem Schutz von
- Daten
- mangelhafter Modularisierung
- und Verteilung von Anwendungen,

Die meisten Probleme werden jedoch durch **Unachtsamkeit** und **Nachsichtigkeit** beim Programmieren verursacht; typische Beispiele hierfür sind:

- Deklarations- und Logikfehler,
- Fehler in Schleifenkonstrukten und conditional expressions,
- unzulängliche oder fehlende Prüfung des Inputs,
- Fehler in Schnittstellen- und Konfigurationsbeschreibungen und sehr häufig
- mangelhaftes Verständnis von elementaren Sicherheitsanforderungen.

Wenn also Fehler in Softwaresystemen einen Großteil von Software-Vulnerabilities ausmachen, so müsste durch eine verbesserte Qualität in der Softwareentwicklung und wenigstens der Reduzierung von „typischen" Fehlern, eine deutliche höhere Sicherheit von Software erreichbar sein.

Web-Applikationen sind „anders"

Aktuelle Web-Applikationen unterscheiden sich erheblich von früheren Business-Applikationen, die nur für bestimmte Anwender (gruppen) erstellt wurden, mehr oder weniger isoliert von anderen Systemen betrieben wurden und deren Architektur vom Client über das (interne) Netz bis hin zu den beteiligten Server-Systemen vollständig unter der Kontrolle der verantwortlichen Betreiber standen. Globalisierung und Wettbewerb zwingen Unternehmen zu Online-Business und extensiver Nutzung der Möglich-

© GAI NetConsult GmbH Nachdruck nur mit Genehmigung des Herausgebers

keiten des Internets in einer Weise, wie sie zu Beginn der Internetnutzung nicht vorhersehbar war. Auch die Geschäftsprozesse in und zwischen den Unternehmen ändern sich heute in rasanter Geschwindigkeit. Ständig müssen die unterstützenden (Software) Systeme nachjustiert, verändert, erweitert oder gänzlich neu erstellt werden. Web-Applikationen wachsen organisch; hohe Produktivität von Entwicklungstools begünstigt das unkoordinierte (und urspr. sogar ungeplante) Entstehen und Wachsen sehr komplexer Anwendungen mit Web-Schnittstellen.

- Immer mehr Software- Komponenten, Code-Fragmente bis hin zu vollständigen Funktionsbausteinen sind frei verfügbar und werden „einfach benutzt" – teils aus Bibliotheken der Entwicklungsumgebungen / Frameworks oder aus dem Fundus der im Internet zugänglichen Bibliotheken für bestimmte Sprachen oder aus dem Umfeld von OSS (Open Source Software).
- Pilotprojekte gehen nach wenigen, zusätzlichen Maßnahmen online, ehemals isolierte und eigenständige Anwendungen werden mit Hilfe heute verfügbarer Tools „web-enabled" und können über das Internet benutzt werden. Diese Systeme wurden aber nie für diesen Zweck konzipiert und es wurden vmtl. auch keine besonderen (Sicherheits-) Vorkehrungen getroffen.
- Selbst Web-Applikationen, die „nur für den internen Gebrauch" konzipiert und entwickelt wurden, mutieren nach und nach zu extern verfügbaren Web-Applikationen – ob durch den remote Zugang per VPN oder durch die (interne) Integration mit anderen Anwendungen, die selbst wieder mit externen Stellen kommunizieren.
- Web-Applikationen basieren auf (Netz- und System) Technologien / Protokollen, die für flexible und ballastfreie („lightweight") Kommunikation geschaffen wurden – nicht für gesicherte Transaktionen auf unsicheren Leitungswegen des Internet, nicht für mission-critical Web-Applikationen.
- Web-Applikationen basieren auf dem Web-Browser mit seinen GUI-Elementen als Front-end; er gehört nicht zur Business-Software, wurde auch nicht vom Anbieter erstellt und kann nicht von diesem kontrolliert werden. Der Browser gilt jedoch weder als zuverlässig (im Sinne von nachgewiesener Stabilität, Korrektheit und Qualität), noch gilt er als sicher – im Gegenteil!

Auf all diesen unsicheren Komponenten baut das Gesamtwerk Web-Applikation auf - können Web-Applikationen überhaupt sicher sein?
Viele Web-Applikationen sind längst produktiv, wenn einzelne Schwachstellen / Angriffsmöglichkeiten entdeckt bzw. bekannt werden. Aber selbst wenn diese bekannt sind, werden sie nicht immer wahrgenommen oder in Beziehung zu den eigenen Programmen gebracht und daraufhin überprüft.
Nicht selten führt die Diskussion um Sicherheitsprobleme aufgrund von Softwarefehlern zu der Behauptung, dass Fehler in komplexen Softwaresystemen irgendwie unvermeidlich sind und gefährdete Systeme von außen geschützt werden müssen, z.B. durch **WAFs** (Web Application Firewalls). Macht es also überhaupt Sinn, kann man das Ziel, sichere Web-Applikationen zu erstellen, überhaupt erreichen? Lohnt es sich, den Aufwand (Organisation, Schulung, Zeit, Kosten) in der Entwicklung zu erhöhen oder sollte man stattdessen besser in Systeme investieren, die jede Web-Applikation vor Angriffen schützt?

Sicherung von Web-Applikationen – therapieren von kranken Patienten?

Für die meisten – oftmals als „mission-critial einzustufenden – Web-Applikationen kommen sämtliche (Sicherheits-)Appelle an die Entwickler bzw. deren Verantwortliche zu spät, denn ansonsten müsste im Zweifelfall sofort der Betrieb eingestellt und der Entwicklungsprozess soweit zurückgesetzt werden, dass eine zuverlässige Überarbeitung des Codes vorgenommen werden kann.
Bei aktuell gezählten knapp 240 Millionen mit mindestens **75 Millionen aktiven Websites** existieren Unmengen von Code in unterschiedlichsten Web-Applikationen – eine Überarbeitung der Software ist schon weder vom Zeitaufwand noch von den Kosten her machbar. Die meisten dieser Web-Applikationen werden ständig weiterentwickelt oder es entstehen bereits Nachfolgesysteme. Alles ist in Bewegung und WAFs sollten dort einsetzen, wo Ursachenbekämpfung zu spät kommt und es um eine Vermeidung oder Reduzierung von Folgen geht.

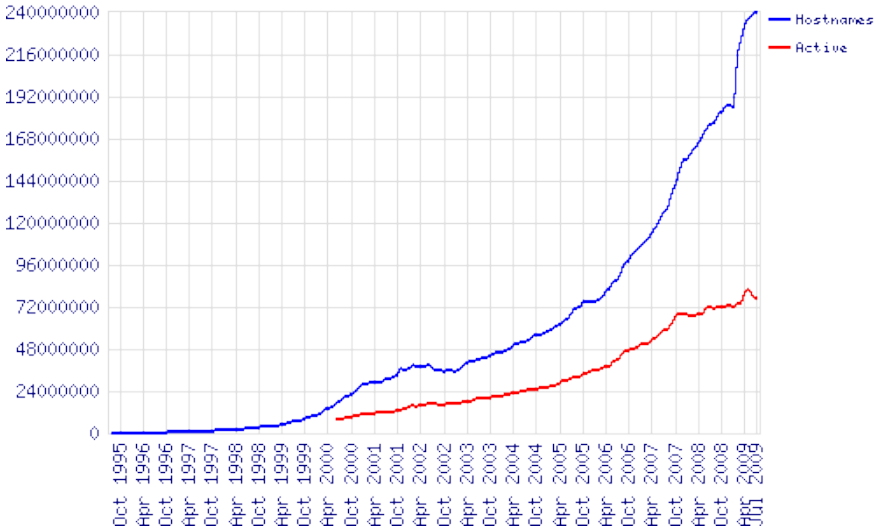

Abbildung-1: Wachstum weltweiter Websites (Quelle: Netcraft)

„penetrate, shield and patch"-Strategie

Aktuelle WAF-Lösungen eignen sich für das Erkennen und Abfangen von:

- • Injectionx Angriffen
  - SQL Injection
  - Command Injection
  - LDAP Injection
  - Skript Injection
  - XPath Injection
- Cross-Site Scripting (XSS)
- Hidden Field Tampering
- Parameter Tampering
- Cookie Poisoning
- Pufferüberlauf¬angriffe
- Forceful Browsing
- Unberechtigten Zugriffen auf Web-Server und
- bekannte Verwundbarkeiten von Web-Anwendungen.

WAFs untersuchen eingehenden und ausgehenden http-Verkehr und entscheiden je nach eingestellter Policy zwischen

- durchlassen,
- durchlassen mit Protokollierung,
- blockieren oder
- Reset der Verbindung.

Wie bei einer normalen Firewall hängt die Effektivität der WAF ab von der Güte der spezifischen Regeln. Neben bereits mitgelieferten und sofort verfügbaren Mustern / Signaturen z.B. bekannte SQL Injection Angriffe, können vor Installation **whitelist Muster** generiert und abgespeichert werden, die durch Auditing der normalen Nutzung der Web-Applikation aufgezeichnet wurden. Abweichungen im realen Betrieb werden dann je nach Einstellung geblockt oder protokolliert.

Vorteile durch Einsatz von WAFs

- Es wird eine weitere Ebene des Schutzes eingeführt, zusätzlich zu den ggf. bereits in der Anwendung vorhandener Filtern.
- Sicherheitslücken können gleichzeitig für mehrere Anwendungen hinter der WAF geschlossen werden, was zu einem schnelleren und – vorerst – kostengünstigeren Schutz führt.
- WAFs können Anwendungen schützen, die mangels Zugriff auf den Quellcode nicht mehr aktualisiert werden können – wenn z.B. die Entwickler (Firma) nicht mehr verfügbar ist.
- WAFs gewähren den Entwicklern einen Zeitgewinn und schützen verwundbare Web-Applikationen solange, bis diese überarbeitet wurden. Dies ist besonders nützlich für neu auftretende Angriffsmuster, die den Entwicklern zum Zeitpunkt der Entwicklung noch gar nicht bekannt sein konnten.

Eine WAF würde also auch dann Sinn machen, wenn die zu schützende Web-Applikation nachweislich keine einzige (heute bekannte) Schwachstelle aufweisen würde.

Nachteile und Risiken der Einsatzmöglichkeit von WAFs

- Web Application Firewalls sind Experten-Tools – Ihre Inbetriebnahme und die regelmäßige Re-Konfiguration sollte nur von gut ausgebildeten Administratoren oder erfahrenen Sicherheitsspezialisten begleitet werden. Der „konservative" Ansatz, die WAF im „block-Mode" mit Aktivierung aller installierter Default-Regeln zu betreiben, wird i.A. zu vielen „false positives" führen, die (gewollte) Funktionalität der Web-Applikation verhindern und auch ohne einen Angriff von außen in einem Desaster im Betrieb enden. Auch falsch konfigurierte Filter können zur Störung des Betriebs führen.
- Wenn die Möglichkeit besteht, die WAF zu umgehen (falsche Konfiguration im Netz), könnten vorhandene Schwachstellen weiterhin ausgenutzt werden.
- Die inhaltliche Prüfung aller eingehenden und ausgehenden Requests kann das Antwortzeitverhalten negativ beeinflussen. Dies gilt umso mehr, wenn der überwiegende Traffic per HTTPS erfolgt und die WAF auch als SSL-Proxy fungiert.
- Die Konfiguration der WAF-Regeln kann nur aktuell bekannte Muster berücksichtigen. Die Einstellungen sollten anhand von aktuellen Erkenntnissen (z.B. anhand der OWASP Top10 oder SANS Top10) bzw. Ergebnissen konkreter Scans und Angriffssimulation regelmäßig durch Fachleute überprüft werden.
- Der Einsatz einer WAF kann zu Unachtsamkeit bei der Entwicklung der Anwendung verleiten – eine WAF ist kein Ersatz für eine sichere Anwendung. Die Strategie von „penetrate-shield-and-patch" führt stets nur zur Behebung, oftmals sogar nur zur **Milderung der Symptome** von Schwachstellen. Der Vergleich mit der generellen Kritik im Gesundheitswesen erscheint angebracht: Dort wird vornehmlich Aufwand in die Behandlung von bereits aufgetretenen Erkrankungen und deren Begleiterscheinungen investiert, als in die Ursachenbekämpfung – sprich: in die Gesunderhaltung.

Entwicklung robuster (sicherer) Web-Applikationen

Wie zu Beginn des Artikels dargelegt, sind viele – wenn nicht sogar die meisten - Sicherheitsprobleme bei Web-Applikationen auf Programmierfehler zurückzuführen. Dies obwohl sich seit Jahren viele Fachleute mit dem Thema Software-Engineering beschäftigen, um den Prozess der Entstehung eines Programmes über alle Phasen qualitativ zu verbessern.

Ist es nicht möglich, fehlerfreie und sichere Web-Applikationen zu entwickeln? Oder ist es zu teuer? Dauert es zu lange? Oder muss Fehlerfreiheit im Kontext von Sicherheit neu definiert werden?

© GAI NetConsult GmbH Nachdruck nur mit Genehmigung des Herausgebers

Fehler-Management während des Softwareentwicklungs-Prozesses

Um während der Entwicklung Fehler zu vermeiden, müssen Fehlermessung und Fehlerbeseitigung gleichermaßen behandelt werden. Dabei müssen Messwerte definiert und angestrebt werden für die erreichte (und zu diesem Zeitpunkt angenommene) Fehlerfreiheit. Nur bei Erreichung dieses Ziels darf innerhalb des Entwicklungsprozesses in die nächste Phase übergegangen werden. Anhand eines einfachen Beispiels soll die Bedeutung von Messwerten und der Zeitpunkt für Messpunkte im Gesamtprozess erläutert werden:

Ein Unternehmen gibt den Entwicklern vor, dass pro **1 Million Lines of Code weniger als 1 Vulnerabilitiy** enthalten sein darf - wie kann man das messen, wie dieses Ziel erreichen?

Angenommen, 25% aller Softwarefehler führen zu Sicherheitsproblemen; daraus folgt als Ziel für die Qualitätskontrolle, dass weniger als 4 Fehler pro 1 Million Lines of Code existieren dürfen.

Wie allgemein üblich, wird das betreffende Unternehmen einen systematischen Fehlertest vmtl. erst am Ende des Entwicklungsprozesses (vor Freigabe der Programme) vornehmen. Da kein Testverfahren alle Fehler finden wird und 100% fehlerfreie Software eher unwahrscheinlich ist, verbleiben bei einer angenommenen Trefferquote von 50% tatsächlich gefundener Fehler in etwa dieselbe **Anzahl versteckte Fehler** im Code. Werden z.B. auch nur 40 Fehler pro 1 Million Lines of Code gefunden (was extrem wenig ist!) und diese alle beseitigt, dann bleiben trotzdem 40 Fehler übrig. Unter o.g. Annahme bedeutet dies, dass 10 Vulnerabilities (=25%) pro 1 Million Lines of Code im als fehlerfrei getesteten und ausgelieferten Code enthalten sind.

Das Qualitätsziel würde hier nicht erreicht – mehr noch, zu diesem Zeitpunkt hat das Unternehmen kaum weitere Optionen, diesen Zustand zu verbessern, es sind schließlich alle (gefundenen) Fehler beseitigt worden!

Die angenommenen Quoten mögen in konkreten Projektsituationen variieren, das Schema der Überprüfbarkeit aber bleibt identisch.

Würden jedoch statt eines einzigen Prüfpunktes ganz am Ende des Entwicklungsprozesses, mehrere und schon viel früher einsetzende Prüfpunkte etabliert und auch hier nur eine Trefferquote von 50% erzielt werden können, ergäbe sich durch die Kaskadierung der Wahrscheinlichkeiten rein mathematisch am Ende weniger Vulnerabilties.

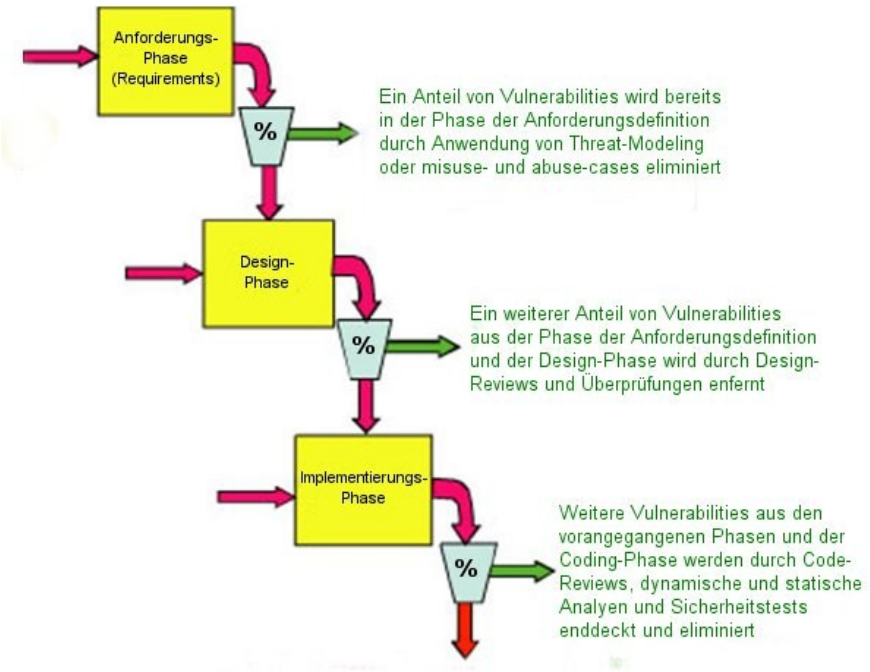

Abbildung-2: Über den Entwicklungsprozess kaskadierter Filter für Vulnerabilities

Fehlerreduktion erhöht die Sicherheit und senkt die Kosten

Nachfolgende Grafik zeigt die schon längst bekannte Erkenntnis, dass es stets teurer wird, erst am Ende des Entwicklungsprozesses erkannte Fehler zu beseitigen.

Im Kontext der Sicherheit von Web-Applikationen geht es aber nicht mehr nur um „normale" Störungen im Betrieb, etwa durch Ausfälle oder fehlerhafte Berechnungsvorschriften, sondern auch um – zunächst – immaterielle Kosten für

- Negativ-Schlagzeilen in der Presse,
- Kundenunzufriedenheit,
- Vertrauensverlust und Verlust von Kunden bis hin zu
- Rechtsstreitigkeiten.

Bei letztem Punkt sind hier nicht nur Forderungen aus Zivilrechtsklagen, sondern – durch die Verschärfung gesetzlicher Bestimmungen im Datenschutz oder **KonTraG** (Gesetz zur Kontrolle und Transparenz im Unternehmensbereich) –auch strafrechtliche Haftungsrisiken zu befürchten. Hieraus kann man ableiten, dass die gezielte Investition in den Software-Entwicklungsprozess nicht nur Kosten spart, sondern für das Unternehmen und seine Repräsentanten von existenzieller Bedeutung ist, fehlerfreie und sichere Web-Applikationen erstellen zu lassen.

Informationssicherheit als Spezialisierung von Funktionssicherheit

Während die Spezifikation einer Web-Applikation im Wesentlichen die Funktionalität des zu erstellenden Systems beschreibt und lediglich im Rahmen der Architekturfestlegung auch Fragen zur Betriebssicherheit (Verfügbarkeit, Performance) und des Zugriffs- und Datenschutzes adressiert wurden, muss zukünftig das Thema Sicherheit **in allen Phasen des Lebenszyklus der Softwareentwicklung** Berück-

© GAI NetConsult GmbH Nachdruck nur mit Genehmigung des Herausgebers

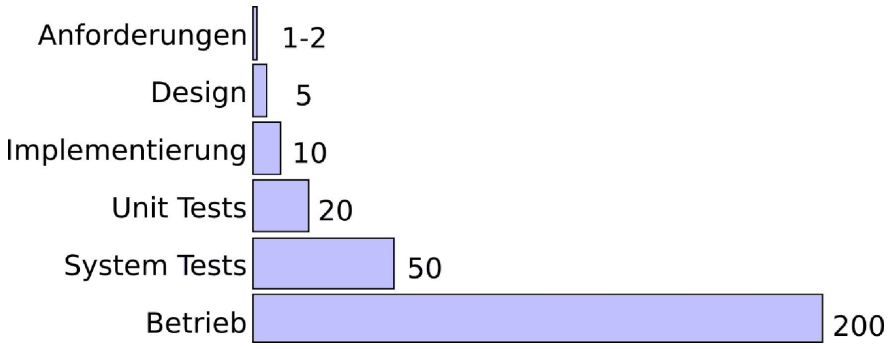

Abbildung-3: Kosten von Softwarefehlern, abhängig von der Projektphase (nach Stuart R. Faulk 1995)

sichtigung finden. Die elementaren Schutzziele

- Vertraulichkeit,
- Unversehrtheit,
- Authentizität,
- Verbindlichkeit,
- Verfügbarkeit und
- Anonymität

müssen bereits als Anforderungen in die Konzeption für jede Komponente festgelegt werden. Im Zusammenhang mit der Bedeutung, die eine Web-Applikation hat oder haben kann, wandelt sich der den LifeCycle begleitende und steuernde Prozess vom **Software-Engineering** zum **Security-Engineering**. Bisher hat sich das Software-Engineering vornehmlich darauf konzentriert, dass eine vorgegebene Spezifikation **unter vorhergesehenen Bedingungen erfüllt wird.** Security-Engineering will zusätzlich verhindern, dass definierte Schutzziele **unter unvorhergesehenen Bedingungen verletzt werden** (z.B. durch Angreifer). In der Öffentlichkeit wird überwiegend über die nach Fertigstellung und Freigabe im Betrieb von Web-Applikationen auftretenden oder möglichen Vulnerabilities diskutiert. Es erscheint einleuchtend zu sein, dass mögliche Risiken so früh wie möglich, spätestens in der Konzeptionsphase, aber besser noch in der Phase der Anforderungsdefinition selbst, adressiert werden.

Der frühe Vogel fängt den Wurm

Software-Architekten und -Entwickler sind (normalerweise) keine Sicherheitsfachleute. Sie sehen ihre Hauptaufgabe darin, die (funktionalen) Anforderungen des Auftraggebers in angemessenem Zeit- und Budgetrahmen umzusetzen. Bis auf wenige Ausnahmen, z.B. Auftraggeber im Öffentlichen Sektor oder in bestimmten Branchen oder Industriezweigen sind Sicherheitsanforderungen nicht klar und eindeutig definiert; hier steht die Erfüllung der funktionalen Anforderungen im Vordergrund. Beispiel für die explizite Anforderung, gewisse Sicherheitskriterien zu erfüllen, gibt es im Finanz- und Kreditsektor. Seit 2007 ist z.B. für alle Vertragspartner der Kreditkartenunternehmen der so genannte **„Payment Card Industry Data Security Standard" (PCI-DSS)** verpflichtend, jedes Unternehmen, das mit Kreditkarten-Daten umgeht, ist für den Schutz der Daten verantwortlich bzw. kann verantwortlich gemacht werden. Im Energiesektor gibt es z.B. durch die **Vorgaben der Bundesnetzagentur** die Verpflichtung zum verschlüsselten Datenaustausch oder zur Compliance im Rahmen des Gesetzes zur Gleichbehandlung aller Netzbetreiber mit Folgen für die eingesetzten IT-Systeme. In allen anderen Bereichen gilt es, den Kunden oder Auftraggeber mit in die Pflicht zu nehmen - nicht zuletzt, um späteren Auseinandersetzungen über die vom Kunden ggf. erwarteten – aber nicht definierten Sicherheitsvorstellungen zu begegnen. **Sicherheit ist bereits bei der Anforderungsdefinition** zu adressieren. So, wie gewollte Verhaltensmuster festgeschrieben werden (**use cases**), sollten ungewollte oder Missbrauchsmuster der Software (**abuse cases**) identifiziert werden.

Design-Prinzipien für sichere Web-Applikationen

Die Qualität von Software im Sinne von Sicherheit kann nur dann erreicht werden, wenn das Thema Sicherheit von Anfang an im Bewusstsein aller am SDLC (Software Development Life Cycle) beteiligten Personen verankert ist.
Peter Löhr (FU Berlin) spricht hier von **„Sicherheitsbewusster Softwaretechnik"** und Ross John Anderson (University of Cambridge) definiert: **„Security-Engineering behandelt Werkzeuge, Prozesse und Methoden für den Entwurf, die Implementierung und den Test von sicheren IT-Systemen."** Die nachfolgende Übersicht basiert auf den von Saltzer und Schröder bereits 1975 veröffentlichten **„8 Prinzipien zur Entwicklung sicherer Systeme"** und sollte auch heute als Guideline für den Entwurf von sicheren Web-Applikationen herangezogen werden.

1 Economy of Mechanism (Einfachheit der Schutzmechanismen)

Komplexes Design erhöht die Fehleranfälligkeit und zugleich auch den Aufwand beim Testen, was dazu führen kann, dass potenzielle Fehlerquellen unentdeckt bleiben.

2 Fail-safe Defaults (Minimale Berechtigungsvergabe)

Zugriffe sollten standardmäßig verhindert, abgelehnt werden. Input-Validierung muss durch Whitelists erfolgen (alles, was nicht erlaubt ist, wird abgelehnt).

3 Complete Mediation (Vollständige Berechtigungsüberprüfung)

Berechtigungsüberprüfung bei jedem Zugriff auf ein Objekt. Vorsicht bei der Optimierung von Zugriffsschutzmechanismen; der Programmierer sollte stets hinterfragen:

- Kann den Ergebnissen bereits

© GAI NetConsult GmbH Nachdruck nur mit Genehmigung des Herausgebers

getätigter Berechtigungschecks vertraut werden?
- Gibt es eine Garantie für die Authentizität der gestellten Anfrage?

4 Open Design (Offener Entwurf)
Die Sicherheit eines Systems darf nicht auf Geheimhaltung des Designs oder der Implementierung beruhen („Kerckhoff's Prinzip": Sicherheit kryptischer Verfahren basiert nur auf Geheimhaltung des Schlüssels, nicht des Verschlüsselungsverfahrens!).

5 Separation of Privilege (4-Augen-Prinzip)
Ein Schutzmechanismus mit zwei Schlüsseln für den Zugang ist robuster als einer, der nur mit einem Schlüssel abgesichert ist. Zusätzlich: Schichten einführen und Rechte nur für jeweils eine Schicht gewähren.

6 Least Privilege (Minimum an Rechten)
Programme und Benutzer erhalten nur die Rechte, die zur Erfüllung der Aufgabe notwendig sind („need to know"); dies hilft im Angriffsfall Auswirkungen und Schäden zu begrenzen.

7 Least Common Mechanism (Minimum an gemeinsamen Mechanismen)
Vermeiden von gemeinsame Variablen / Datenbereichen. Funktionen und Ressourcen nicht gleichzeitig mehreren Benutzern zur Verfügung stellen.

8 Psychological Acceptability (Psychologische Akzeptanz und Usability)
Sicherheitsmaßnahmen müssen benutzertauglich sein (Akzeptanz und Handhabbarkeit); Einfachheit fördert allgemeine und routinemäßige Benutzung, komplizierte Verfahren werden verweigert oder unterlaufen („Zettelchen" mit kryptischen Passwort unter der Tastatur).

Tools zur Security-Unterstützung im Entwicklungsprozess
Mit Hilfe von **statischen Quellcode-Analyzern** kann der Code auf möglicherweise unsichere Bibliotheksaufrufe gescannt werden. Eine weitere spezifische Testmethode z.B. kann Fuzzing sein. Mit Fuzzing-Tools – oder auch **Robustness Testing** oder **Negative Testing** - werden automatisch zufällige Daten erzeugt, die über Eingabeschnittstellen eines Programms verarbeitet werden. Fuzzing wird in der Regel im Rahmen eines Black-Box-Tests durchgeführt, wodurch gerade Web-Applikationen sehr einfach getestet werden können. Bei festgestellten Problemen oder um im Kontext der konkreten Web-Applikationen inhaltlich tiefer zu testen, muss auf **White-Box-Testing** übergegangen werden, womit z.B. ein ablaufbezogenes Testen möglich ist, bei dem der Ablaufgraph im Vordergrund steht. Ziel des Tests ist es, sicherzustellen, dass Testfälle in Bezug auf die Überdeckung des Quellcodes minimalen Kriterien stand halten, wie beispielsweise

- Zeilenüberdeckung:
  Ausführung aller Quellcode Zeilen,
- Anweisungsüberdeckung:
  Ausführung aller Anweisungen,
- Zweigüberdeckung bzw. Kantenüberdeckung:
  Durchlaufen aller möglichen Kanten von Verzweigungen des Kontrollflusses,
- Bedingungsüberdeckung bzw. Termüberdeckung:
  Durchlaufen aller möglichen ,ausschlaggebenden Belegungen bei logischen Ausdrücken in Bedingungen,
- Pfadüberdeckung (mehrere Varianten):
  Betrachtung der Pfade durch ein Modul.

Ferner sind mit Hilfe von spezialisierten **Security- und Penetrationsscannern** dynamische Tests von Komponenten oder der ganzen Web-Applikation vorzunehmen. Wichtig dabei ist, dass die Tests sowohl ohne Zugriffsrechte, als auch mit Gewährung definierter Zugriffsrechte erfolgen; oftmals ist in der Praxis ein autorisierter Benutzer selbst der Angreifer bzw. wird sein Zugang von diesem missbraucht.

Thematisieren von Sicherheit im gesamten Software-entwicklungs-Prozess
Es kann zwar die Fehlerreduzierung wesentlicher Faktor für die Verringerung der Verwundbarkeit von Web-Applikationen sein – sie allein führt jedoch nicht automatisch zu sicheren Web-Applikationen. Es ist enorm wichtig, allen Beteiligten im Entwicklungsprozess die Zusammenhänge und Ursachen für Vulnerabilities verständlich zu machen. Zu oft werden den Entwicklern die häufigsten Ursachen nur in Form einer abzuarbeitenden Liste vorgegeben (z.B. Vermeiden / Verhindern von Buffer-Overflows, SQL-Injection, Cross-Site-Scripting oder Race conditions). Es ist jedoch auch das Verständnis und die Denk- bzw. Handlungsweise eines potenziellen Angreifers und die Wirkungsweise bzw. Abhängigkeiten von Ursachen in der Software(Lösung) zu vermitteln. Wichtig ist der Know-how-Erwerb, **wie** diese möglichen Schwachstellen ausgenutzt werden können, damit sich nach und nach im gesamten (Denk) Prozess das Thema Sicherheit wie selbstverständlich verankert.
Die folgende Abbildung zeigt einige best-practice-Ansätze auf, mit denen das Thema Sicherheit in den elementaren Phasen des Entwicklungsprozesses thematisiert werden kann, ohne hierbei einen schematischen Ablauf vorzugeben. Da Zeitdruck in den Projektteams und das Fehlen von (in Sicherheitsfragen) erfahrenen Projektleitern starken Einfluss auf die best-practice-Regeln haben kann, ist eine übergreifende organisatorische Unterstützung des Managements bei der Einstellung von Security-Policies, deren Überwachung sowie für entsprechende Trainingsmaßnahmen erforderlich. Es ist wichtig, dass die Projektleitung die Planung und Durchführung der festgelegten Sicherheitsmaßnahmen überwacht. Ein gleich zu Beginn aufgesetztes Risiko-Management ist erforderlich, damit alle eventuellen Bedrohungen frühzeitig identifiziert, zugeordnet und bearbeitet werden können. Aufgrund der schon vor Jahren enorm gestiegenen Angriffe auf die verbreiteten Microsoft-Produkte und die deshalb be-

© GAI NetConsult GmbH Nachdruck nur mit Genehmigung des Herausgebers

kannt gewordenen Schwachstellen hat der Hersteller schon Ende 2004 den **„Trustworthy Computing Security Development Lifecycle"** definiert, der als Muster für einen sicheren Softwareentwicklungsprozess gedacht ist. Microsoft benutzt das Akronym **SDL** für „Security Development Lifecycle", stützt sich dabei aber auch auf vorab definiert Grundsätze zur Erstellung sicherer Software, die als SD3 bezeichnet werden:

- Security by Design – für sicherheitsgemäße Planung, Entwurf und Implementierung.
- Security by Default – alle Start-up-Einstellungen und Berechtigungen müssen auf minimale Rechte gesetzt werden, weil immer angenommen werden kann, dass die Software angreifbar sein könnte.
- Security in Deployment – die Auslieferung freigegebener Software inkl. von späteren Updates muss durch automatisierte Verfahren und mit ausführlichen Dokumentationen unterstützt werden.

Microsoft ergänzt diese Liste durch

- Communications – Softwareentwickler sollten auf die Entdeckung von Schwachstellen vorbereitet sein, offen mit Anwendern über mögliche Risiken kommunizieren und aktiv bei geeigneten Schutzmaßnahmen behilflich sein

und führt eine Erweiterung dieser Prinzipien als **SD³+C** ein.

Die hierzu erarbeiteten Dokumente sowie aktive Unterstützungen in der Entwicklungsplattform sind von Microsoft kostenlos erhältlich und z.B. als SDL Process Template für die Visual Studio Team System (VSTS) Software-Entwicklungsumgebung verfügbar.

Fazit – wie in der Medizin: Therapieren der Kranken, schützen der Anfälligen und den Gesundheitsprozess fördern

Das vergleichende Beispiel der Praxis in der medizinischen Versorgung scheint auch nach Betrachtung der unterschiedlichen Ansätze anwendbar, wenn nicht sogar notwendige Maßnahme zu sein: Es gibt zu viele bereits infizierte und anfällige Web-Applikationen im produktiven Einsatz oder in der laufenden Entwicklung kurz vor Freigabe – die dort enthaltenen Vulnerabilities müssen abgefangen, mögliche Symptome müssen therapiert werden. Auch in der Zukunft sind geeignete Schutzmaßnahmen (z.B. Shields, WAFs) für fertige Web-Applikationen erforderlich, um den sich weiter verbreitenden und neuartigen Angriffen trotzen zu können. Damit jedoch überhaupt die Chance besteht, in der Zukunft den Trend umzukehren und nach und nach nur noch „gesunde", d.h. sichere und weniger angreifbare Software zu haben, muss massiv in die (Sicherheits-)Ausbildung von Entwicklerteams und das Projektmanagement investiert werden. Es stellt sich also nicht die Frage, Einsatz von WAFs **oder** verbesserte Entwicklungsprozesse für sichere Web-Applikationen – der Markt hat Bedarf für beides. Der entscheidende Erfolgsfaktor wird darin bestehen, die Synergie zwischen Experten-Know-how von Sicherheitsspezialisten und erfahrenen Softwarearchitekten zu fördern.

Referenzen:

[Jones] Jones, Capers. "Software Assessments, Benchmarks, and Best Practices".

[Jacquith] Jacquith, Andrew. „The Security of Applications: Not All Are Created Equal."

[Whitepaper Qualys] „Building a Web Application Security Program"

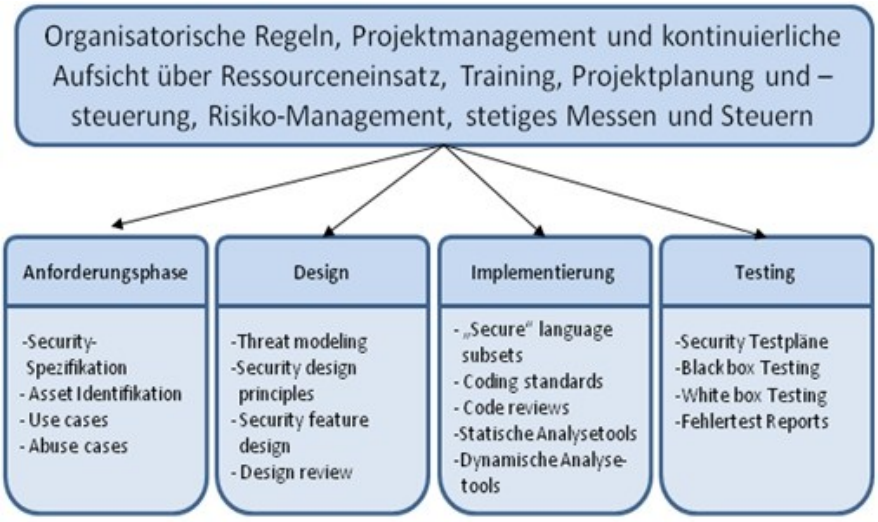

Abbildung-4: Adressieren von Sicherheit während des Softwareentwicklungs-Prozesses

Die **GAI NetConsult GmbH** konzentriert sich als System- und Beratungshaus auf die Planung und Realisierung von sicheren eBusiness Lösungen. Dabei wird der gesamte Prozess von der Analyse über die Konzeption und Realisierung bis zur Überwachung angeboten.

WEITERE INFORMATIONEN

EIN SERVICE DER

GAI NetConsult

FÜR IHR ABONNEMENT BESUCHEN SIE BITTE UNSERE WEB-SEITE

www.gai-netconsult.de/