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

Standards und Verfahren für mehr Sicherheit im Software-Entwicklungsprozess (Teil 1)

Wilfrid Kettler (GAI NetConsult GmbH)

August 2010 Sonderdruck aus Security *Journal* #50

Technik ersetzt keine Organisation – so oder so ähnlich wird die Erkenntnis auch im Bereich der Softwareentwicklung lauten: so wenig, wie die Auswahl einer restriktiven Programmiersprache zwangsläufig zu einem sicheren Programm führt, so wenig kann z.B. der Einsatz von SSL das Kompromittieren von Web-Applikationen verhindern. Dazu muss mehr gehören als Technik und Werkzeuge. Dieser Artikel wird sich folgerichtig weniger mit dem Einsatz neuer (Scanning-) Werkzeuge, Input-Filter, Hash- oder Verschlüsselungstechniken befassen, sondern versucht einen Querschnitt über die Anstrengungen von staatlichen Behörden oder ihnen nahe stehenden Institutionen, Non-Profit-Organisationen oder Software-Herstellern und IT-Dienstleistern zu geben, nachhaltig Qualität und Sicherheit in den Prozess der Software-Entwicklung zu integrieren.

Es werden zwar im Zusammenhang mit Sicherheitsproblemen bei der Software zumeist die Entwickler „an den Pranger" gestellt, aber ohne den organisatorischen Überbau und die notwendige Managementunterstützung kann sich wenig ändern. Insofern dient dieser Artikel auch mehr als Anregung für Verantwortliche, sich Hilfestellungen für die notwendige organisatorische Ausrichtung und bewährte Verfahrensmuster zu beschaffen.

Das Problem mit nicht-funktionalen Anforderungen an Software

Auf der einen Seite dürfte die Sicherheit von Software als elementares Merkmal des Qualitätsbegriffes ausreichend anerkannt sein und von daher schon längst in entsprechende Vorgehensmodelle und Qualitäts-"Offensiven" des Software-Entwicklungsprozesses Eingang gefunden haben. Tatsächlich jedoch fällt es Entwickler(Teams) immer wieder schwer, die Anforderung nach Erstellung „sicherer Software" zu erfüllen.

Die funktionalen Anforderungen eines Softwaresystems sind i.d.R. im Lastenheft des Auftraggebers formuliert und werden im Fachkonzept entsprechend spezifiziert. Daher sind diese Anforderungen leicht nachvollziehbar und werden im Entwicklungsprozess auch entsprechend umgesetzt und sind in Test- und Abnahmeprozeduren leicht nachzuprüfen bzw. ist deren Einhaltung leicht nachweisbar. Eine Reihe nicht-funktionaler Anforderungen, wie z.B. :

- Antwortzeiten unter bestimmten Nutzungsbedingungen,
- Verfügbarkeiten / Bereitstellungszeiten,
- Archivierungsfristen oder
- Sicherstellung der Vertraulichkeit gespeicherter Daten (z.B. durch Verschlüsselung)

werden durchaus ebenfalls standardmäßig in den Entwicklungsprozess eingebaut bzw. dort berücksichtigt. Weitergehende Maßnahmen, die eine Abwehr von potenziellen Bedrohungsszenarien einschließen, sind dagegen (leider) nur selten offizieller Standard und im Zweifelsfall auch nicht **standardisiert**, d.h. gehören heute nicht zum Standard-Repertoire eines Software-Entwicklers – wird oder wurde dies also eventuell nicht gelehrt?

Schon lange gehören einzelne Lehrveranstaltungen aus dem Umfeld der Informationssicherheit zum Standardumfang eines Informatik-Studiums in Deutschland (Kryptographie, Datenschutz etc.). Vielfach gibt es auch spezielle Lehrveranstaltungen zum Thema IT-Sicherheit, in denen verschiedenste Themen gebündelt werden. Ob jedoch das Thema Sicherheit bereits als fester Bestandteil innerhalb des Software-Entwicklungsprozesses gelehrt wird, muss angesichts der Situation am Softwaremarkt bezweifelt werden.

Immerhin gibt es an mittlerweile sieben deutschen Hochschulen oder Fachhochschulen ausgewiesene **Bachelor- oder Master-Studiengänge zur IT-Sicherheit**, neben der Ruhr-Universität Bochum als Vorreiter (seit 2001), der Hochschule Aalen, FH Brandenburg, FH Gelsenkirchen, FH Offenburg, Uni Passau und ab dem Wintersemester 2010 auch an der TU Darmstadt (Anmer-kung: Diese Aufzählung hat keinen Anspruch auf Vollständigkeit).

Allerdings besteht nirgends Zweifel darüber, dass Sicherheit als Qualitätsmerkmal begriffen wird und es unverzichtbar ist, zu verschiedenen Zeitpunkten des Entwicklungsprozesses die Qualität messen und bewerten zu können.

Um Qualitätsverbesserungen in Prozessen – gleich welcher Art – erzielen zu können, gibt es diverse Modelle, Vorgehensweisen und Standards. So werden z.B. auch im Rahmen von Software-Engineering-Veranstaltungen Modelle internationaler Standardisierungsgremien und Organisationen zur Beurteilung des Reifegrades im Entwicklungsprozess behandelt.

Die Idee dahinter – oder einfach die Notwendigkeit zum Handeln

Mittlerweile dürfte bei jedem Einzel-Entwickler von Software bis zum großen Softwareunternehmen die Erkenntnis vorliegen, dass Qualität nicht von selbst oder nebenbei entsteht. Stattdessen ist systematisches Planen, Implementieren und Testen von Nöten und – dies ist möglicherweise in der Realisierung und gegenüber allen Marktteilnehmern schwierig durchzusetzen - Qualität kostet zusätzliches Geld. Einerseits sind Ausbildung und das Vorhalten von dedizierten Testteams teuer, dazu kommt noch eine kostenintensive Ausstattung mit geeigneten Werkzeugen. Werden nicht bereits zu Beginn des Projektplans ausreichende Tests im Zeitplan berücksichtigt oder kommt es durch hohe Fehlerzahlen zu Verzögerungen, hat die Qualitätssicherung Einfluss auf die termingerechte Ablieferung des Produktes. In Konfliktfällen wird nicht selten eher eine unfertige oder nicht genügend – auf Sicherheitsprobleme hin - getestete Software akzeptiert und ausgeliefert, als dass Terminüberschreitungen akzeptiert wurden.

Der zunehmende Wettbewerbsdruck und die Orientierung der Kunden und Auftraggeber, bei Ausschreibungen nach dem „preiswertesten" Angebot zu entscheiden, verschärft diese Situation noch weiter.

Führende Analysten vergleichen die Lage der meisten Software-Unternehmen mit der **Fertigungsindustrie in den 80er Jahren**. Gerade zuvor hatte es die Japanische Industrie vorgemacht, qualitativ hochwertige Produkte zu extrem wettbewerbsfähigen Preisen anzubieten. Unter dem Oberbegriff **„Total Quality Management"** etablierten sich hiernach japanische Managementansätze weltweit als Standards beim Qualitätsmanagement.

Begreift man Software bzw. die Entstehung von Software – im ingenieurwissenschaftlichen Kontext – als Industrieprodukt, so liegt es nahe, auch die Software-Entwicklung hinsichtlich des Qualitätsmanagements weitgehend zu industrialisieren und am Prozessverständnis fertigender Unternehmen auszurichten.

Die vielfältigen Standardisierungsbemühungen hierzu wurden teilweise durch staatliche Initiativen getrieben (nicht selten auch aus dem Umfeld von Militär oder Geheimdiensten). Daneben gibt es eine Reihe entsprechender Initiativen der Privatwirtschaft (z.B. großer Softwarehersteller) sowie freier Träger und Standardisierungsgremien. Auch wenn von den jeweiligen Initiatoren und Autoren nicht selten Versprechungen auf „sicheren Code" erkennbar sind, können diese angesichts der Komplexität und Unterschiedlichkeit individueller Anforderungen nicht als „Evangelium" zur sicheren Softwareerstellung verstanden werden.

Im Folgenden wird – ohne Anspruch auf Vollständigkeit - versucht, einen Überblick zu geben über verschiedene, teilweise unterschiedliche und auch sich ergänzende Ansätze und Bemühungen, um Ordnung sowie Mess- und Prüfbarkeit in den Prozess der Softwareentwicklung zu bringen. Neben einer allgemeinen Verbesserung der Qualität steht dabei auch und immer mehr das Thema Sicherheit im Fokus. Dabei geht es weniger um die umfängliche Darstellung der jeweiligen Programminhalte – diese können leicht bei den Gremien direkt nachgelesen werden. Vielmehr soll aufgezeigt, werden, dass es an vielen Stellen bereits weit reichende Vorüberlegungen bis hin zu Empfehlungen (Best Practices) gibt, die der geneigte Leser – als interessierter Entwickler oder als Verantwortlicher für die sichere Entwicklung – auf seine eigene Situation und individuelle Anforderungen hin adaptieren kann.

Der **Trustworthy Computing Security Development Lifecycle** (Abgekürzt **SDL**, zu Deutsch **Entwicklungszyklus für vertrauenswürdigen Computereinsatz**) ist ein schon 2004 von Microsoft veröffentlichtes Konzept, den Entwicklungsprozess hinsichtlich der Qualität von Software nachhaltig zu verbessern und böswillige Angriffe abwehren zu können bzw. diesen keine Angriffsfläche zu bieten. SDL gilt unternehmensweit verbindlich für alle entwickelten Produkte und soll letztlich durch Kombination eines ganzheitlichen und praxisorientierten Ansatzes zur Verringerung der Anzahl und Schwere von Sicherheitslücken in Software beitragen.

Wesentliche Grundsätze beim SDL sind:

- **Secure by design** - Schon in der Planungsphase sollte auf die Sicherheitsbelange der Software eingegangen werden,
- **Secure by default** - Trotz sorgfältigster Planung sollte ein Entwickler von dem Vorhandensein von Sicherheitslücken ausgehen. Aus diesem Grund sollten die Standardeinstellungen (z.B. erforderliche Privilegien) möglichst niedrig gewählt und selten benutze Features standardmäßig deaktiviert werden,
- **Secure in deployment** - Die mitgelieferten Dokumentationen und Tools sollen die Administratoren dabei unterstützen, die Software möglichst optimal einzurichten.
- **Communications** - Alle Entwickler sollen offen mit möglichen Sicherheitslücken umgehen, mit den Endanwendern kommunizieren und ihnen schnellstmöglich Patches oder Workarounds zur Verfügung stellen.

Der Microsoft-SDL ordnet alle

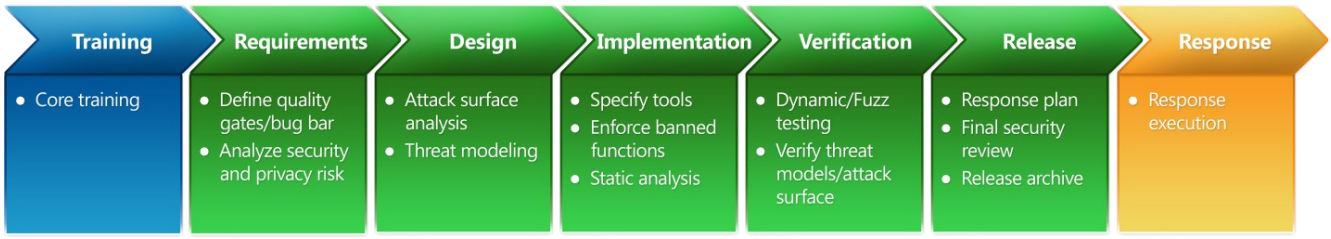

Abbildung-1: Phasen des Software-Entwicklungsprozesses im Microsoft SDL

der Sicherheit dienenden Aktivitäten und Maßnahmen den einzelnen Phasen des Entwicklungsprozesses zu:

SDL wurde zwar für den internen Gebrauch entwickelt, soll aber möglichst allen Entwicklern und Software-Herstellern als Leitfaden und Modell dienen, auch selbst die Qualität und die Sicherheit der erstellten Software zu verbessern. Um das ursprüngliche Konzept auch in kleineren Organisationen anwenden zu können, gibt es das SDL Optimization Model, welches die wichtigsten Verantwortungsbereiche nun in 5 Areas einordnet bzw. zusammenfasst, die grob nach den entsprechenden Phasen innerhalb des Lebenszyklus der Softwareentwicklung ausgerichtet sind:

- **Ausbildung / Training, Richtlinien und Managementunterstützung**

  Hier ist im Wesentlichen für die Awareness aller Projektbeteiligten zu sorgen, sind bestehende und notwendige Regeln zu definieren bzw. zu kommunizieren und mit Blick auf Sicherheitsrisiken gezielte Schulungen durchzuführen. Diese Phase gehört streng genommen nicht zu einem einzelnen Projekt, weil sie die Unternehmensorganisation bis hin zu Verfahren für die standardisierte Verfolgung von Qualitätsmängeln und Sicherheitsproblemen behandelt.

- **Anforderungen und Design**
  Auf der Basis einer individuellen Risikoanalyse ergeben sich Konsequenzen und Maßnahmen bzgl. des Softwareentwurfs; hierfür eignet sich das „Threat Modeling", ein systematischer Ansatz zum Entdecken und Dokumentieren von Bedrohungen und Risiken im Kontext des jeweiligen Anwendungsszenarios schon im Entwurfsstadium sowie entsprechende Security Design Reviews. In dieser Phase sind im Wesentlichen die Projektmanager und Software-Architekten, sowie projektübergreifend etablierte Sicherheitsexperten gefragt.

- **Implementierung**
  Hat man Sicherheitsaspekte im Entwurf berücksichtigt und eingeplant, gilt es nun, die Entwicklung nach den Vorgaben sicherer Programmierung durchzuführen. Unabhängig von der jeweiligen Programmiersprache sind folgende fünf Punkte unbedingt zu beachten:
  - Prüfung von Ein- und Ausgaben
  - Authentisierung und Zugriffskontrollen
  - korrektes Handhaben und Schutz von sensitiven Informationen und Daten
  - Befolgen des "Least Privilege"-Prinzips, das die geringst möglichen Privilegien vergibt
  - Verhindern von Informationspreisgaben

  Durch die Befolgung dieser Prinzipien lassen sich viele Sicherheitslücken in der Implementierungsphase wie SQL-Injection, Cross-Site Scripting oder Fehler in der Zugriffskontrolle vermeiden.

- **Überprüfung**
  Dieser Bereich beschreibt die Vorgehensweisen und Maßnahmen bei der Entdeckung und Behebung von Schwachstellen, die sich bei der Konzeption und Umsetzung eingeschlichen haben könnten. Hierbei finden gezielt Werkzeuge Anwendung, wie z.B. Tools zur dynamischen Analyse oder Fuzzying. Mit ihnen lassen sich Eingabeschnittstellen mit generierten Eingaben beschicken und so lange testen, bis gewährleistet ist, dass das Programm auch bei unerwarteten Eingaben keine Probleme mehr verursacht. Weitere hilfreiche Maßnahmen sind Sicherheitstools (Schwachstellen-Scanner) und manuelle Code-Reviews. Die Verantwortlichkeit dieser Arbeiten obliegt der Qualitätssicherung und den Entwicklern gleichermaßen, wobei jeweils die gezielte Unterstützung eines Sicherheits-Experten hinzuzuziehen ist.

- **Release-und Reaktionssystem**
  In diesem Projektabschnitt werden die Aktivitäten und Maßnahmen beschrieben, um vor Freigabe und Abnahme die Einhaltung der Vorgaben durch ein Final Security Review (FSR) zu überprüfen. An dieser Stelle wird auch entschieden, welche Verbesserungen in

der Reaktionsfähigkeit und Flexibilität der Projektorganisation erforderlich sind, um auf später entdeckte Sicherheitslücken zeitnah und mit möglichst geringen Kosten reagieren zu können; dies schließt interne Aktivitäten genauso ein, wie Reaktions- und Kommunikationsmaßnahmen bei den Kunden, an die bereits Software ausgeliefert wurde.

In Anlehnung an das **Capability Maturity Model (Reifegradmodell)® (kurz CMM®)** ist auch in Microsofts SDL ein Reifegradmodell zur Beurteilung der Qualität (=Reife) des gesamten Softwareprozesses bzw. jeder einzelnen Phase integriert.

Alle Anstrengungen zur Verbesserungen der Qualität und Sicherheit im Softwareentwicklungsprozess werden letztendlich nur dann nachhaltig Erfolg haben, wenn die dazu eingeführten Organisationen, Regeln, Aufgaben und Verantwortlichkeiten bis hin zum Einsatz bestimmter Verfahren und Werkzeuge einer permanenten Beobachtung und Beurteilung unterliegen und stetig verbessert – optimiert – werden.
Während bei CMM die Qualität bzw. der Reifegrad eines betrachteten Bereichs in 5 Stufen bewertet wird, definiert Microsoft 4 verschiedene Optimierungs- bzw. Reifegrade:

1 - basic
Dies ist der Grundzustand, den jede Organisation leicht erreicht, auch ohne dass ein Prozess für die Softwareentwicklung definiert und umgesetzt wird. I.d.R. erfolgen alle Maßnahmen manuell, nur gelegentlich und auch nicht zentral koordiniert. Es gibt hier keine Standards und Richtlinien für Sicherheit, weder bezogen auf die Architektur des zu erstellenden Systems noch auf das Design, die Entwicklung, den Test, die Überprüfung oder auf eingesetzte Sicherheitsmaßnahmen. Existieren dennoch entsprechende Verfahren oder werden Tools eingesetzt, gelten diese nur für ein konkretes Projekt. Mangels einer Systematik ist der jeweilige Zustand unbestimmt; zu erwartende Kosten, Zeiten und Qualität sind nicht vorhersehbar.

2 – standardisiert
Auf dieser Ebene existieren standardisierte Sicherheitsmaßnahmen, die aber gerade erst eingeführt werden. Die Projektorganisationen auf dieser Ebene sind in der Lage, die Sicherheit und deren Bedrohungen bei neuen Projekten zu bewerten und geeignete Mitarbeiter für die Umsetzung der Sicherheitsmaßnahmen und den Datenschutz in der Softwareentwicklung zu bestimmen. Erste Standards und Richtlinien sind definiert, bedürfen aber noch weit reichender Verbesserungen, da viele Maßnahmen nur als Pilotversuch für einzelne Projekte praktiziert werden. Das Management akzeptiert, unterstützt die Maßnahmen zur Verbesserung der Sicherheit aber noch nicht aktiv. Ein Großteil der Anstrengungen setzt erst in (zu) späten Phasen des Entwicklungsprozesses ein und oft erst, nachdem bereits Warnsignale zurückgemeldet wurden; die Folge sind häufige Schwankungen in der Qualität und ungeplante Kosten für Verbesserungsmaßnahmen.

3 - fortgeschritten
Sowohl für das Produkt als auch für den Prozess werden quantitative Ziele vorgegeben, ihre Erreichung gemessen und überwacht; Zeiten, Kosten und Qualität sind zuverlässig kontrollierbar. Hier ist die Managementunterstützung für notwendige Maßnahmen explizit und ausreichend vorhanden und Projekte mit höherem Risikopotenzial werden standardmäßig in die eingespielten Regularien eingeordnet, es gibt keine Ausnahmen. Alle Maßnahmen zur Herstellung von Sicherheit und Datenschutz sind an den richtigen Stellen im gesamten Softwareentwicklungsprozess fest verankert; durch abgestimmte Verzahnung der auf einander folgenden Verantwortungsbereiche wird eine hohe Effizienz und Güte erzielt. Richtlinien für Sicherheitstests und geeignete Werkzeuge zur Unterstützung sind fest in den Teams etabliert, werden gut beherrscht und helfen, Kosten zu reduzieren ohne die Qualität zu gefährden. Die Reaktion auf festgestellte (Sicherheits-) Probleme ist adäquat hinsichtlich Terminen und Koordination. Die routinierte Anwendung des Erlernten hilft gleichermaßen, hohe Qualität zu liefern und dabei Kosten zu senken.

4 - dynamisch
Die Erreichung dieses Reifegrades bedeutet ein strategisches Plus für das entwickelnde Unternehmen. Über die erfolgreich mit hoher Qualität und Sicherheit abgewickelten Software-Projekte hinaus, können diese Unternehmen aufgrund ihrer Erfahrung auch ihre Kunden unmittelbar unterstützen und heben sich gegenüber ihren Mitbewerbern (mit geringeren Reifegraden in den Softwareentwicklungsprozessen) ab. Die gesamte Organisation konzentriert sich auf das Finden von Schwächen und die weitere Verbesserung des Prozesses. Mit einer reichhaltigen Erfahrung und einem systemimmanenten Sicherheitsbewusstsein können aus den Teams strategische Aufgaben abgeleitet und anderen Bereichen des Unternehmens zur Verfügung gestellt werden, z.B. auch die Überarbeitung von Softwareprodukten, die zwar fertig und in Produktion sind, aber noch nicht unter dem neuen Sicherheitsbewusstsein und den bewährten Verfahren entwickelt wurden. Nicht selten – das wird auch aus anderen Erfahrungsberichten beim Einsatz von Reifegradmodellen berichtet – entstehen aus der Kombination erfahrener, sicherheitsbewusster Software-Entwickler und reinen Sicherheitsexperten, spezielle Teams, die in der Lage sind, auch pro aktiv nach Bedrohungsszenarien und potenziellen (noch unerkannten) Schwachstellen forschen.

Die vorgenannten Darstellungen entsprechen im Wesentlichen exakt den (idealtypischen) Formulierungen aus den einschlägigen Microsoft-Publikationen zum Thema Trustworthy Computing. Selbstverständlich ist Microsoft – wie auch anderen „Gurus" aus dem Sicherheitsbereich – bewusst, dass die bloße Formulierung, welche Anstrengungen

man unternehmen muss, nicht zur Erreichung der Ziele ausreicht. Vielfach liegt der Schwerpunkt der Empfehlungen auch auf dem „was" und „an welcher Stelle" im Prozess die Dinge zu verbessern sind; das „wie" das dann in einem einzelnen Unternehmen umgesetzt wird oder werden kann, muss stets individuell ermittelt werden. Der Microsoft SDL versteht sich daher auch nur als Vorschlag zur Strukturierung, als Template. Letzteres wird seit ein paar Jahren sogar explizit praktiziert. Microsoft stellt Entwicklern z.B. kostenlose Tools zum Auffinden von Sicherheitslücken in Anwendungen zur Verfügung. Der BinScope Binary Analyzer überprüft binären Code darauf, ob alle empfohlenen und notwendigen Security Flags, Schutzmechanismen und Kontrollen in der Software vorhanden respektive aktiviert sind. Mit dem MiniFuzz File Fuzzer können Entwickler ihre Anwendungen auf unerwartete Verhaltensweisen testen und so früh im Entwicklungsprozess feststellen, ob etwa Programmabstürze als Sicherheitsrisiken untersucht werden müssen.

Ferner stellt Microsoft sein **Security Development Lifecycle (SDL) Process Template** zum kostenlosen Download auf MSDN bereit. Das Plug-In für Visual Studio Team System integriert den SDL Prozess direkt in die vorhandene Entwicklungsumgebung.

SDL richtet sich auch an Entwickler, die **Agile Softwareentwicklung** (Ansatz, der versucht die Softwareentwicklung flexibler und schlanker zu gestalten) betreiben (SDL for Agile Development). Einer Forrester-Studie zufolge arbeiten bereits 85 Prozent aller Entwickler damit oder sind kurz davor, den Prozess zu nutzen. Der Security Development Lifecyle umschließt somit nun einen deutlich größeren Entwicklerkreis.

Eine Reihe anderer Reifegradmodelle für Qualität und Sicherheit im Software-Entwicklungsprozess haben sich die Erfahrungen und Ergebnisse des Microsoft SDL zunutze gemacht und sind mehr oder weniger vergleichbar. Vielleicht sind diese Alternativen auch erst durch diese Vorarbeit bzw. den Nachweis möglich geworden, dass die konkrete Anwendung beim größten Software-Hersteller der Welt selbst, die geforderten Ansprüche auch tatsächlich erfüllen kann.

BSI – Build Security In

Nicht von einem anderen Software-Hersteller und auch nicht von einer privat-wirtschaftlichen oder Non-Profit-Organisation stammt „Build Security In". Aller Wahrscheinlichkeit nach dürfte zumindest dem deutschen Leser dieses Kürzel bekannt vorkommen – bezeichnet sich doch das „Bundesamt für Sicherheit in der Informationstechnik (BSI)" ebenso.

"Build Security In" ist jedoch eine strategische Initiative zur Software Qualitätssicherung der **National Cyber Security Division (NCSD)** des **US Departments of Homeland Security** und geht zurück auf die National Strategy to Secure Cyberspace, die Präsident W. Bush im Februar 2003 veranlasst hat.

Diese Initiative arbeitet eng mit dem **Software Engineering Institute (SEI)** der Carnegie Mellon University zusammen und hält u.a. themenbezogene Veröffentlichungen namhafter Autoren und Fachleute zur öffentlichen Nutzung bereit. Ziel ist die Erarbeitung und Veröffentlichung von Verfahren, Werkzeugen, Leitlinien, Regeln und Standards als offenes Angebot an alle Software-Entwickler, Software-Architekten und Sicherheitsexperten, um systematisch Sicherheit in jeder Phase der Software-Entwicklung zu integrieren.

Natürlich wurde diese Aufgabe dem US Ministerium nicht ganz uneigennützig übertragen; schließlich gefährden unsichere Programme das Funktionieren der kritischen Infrastrukturen des Landes, öffentliche und private Systeme und Ressourcen eingeschlossen.

Im Wesentlichen zielen die Initiativen zur Software-Qualitätssicherung darauf ab, Sicherheitsschwachstellen in Software zu reduzieren und deren Ausnutzung zu verhindern. Es sollen Wege aufzeigt werden, um den Entwicklungs- und Auslieferungsprozess von sicherer Software zu ermöglichen bzw. weiter zu verbessern. Speziell für die Belange der Software-Qualitätssicherung und Sicherheit in der Softwareentwicklung wird vom SEI die Webseite https://buildsecurityin.us-cert.gov/swa/ gepflegt; hier sind alle Veröffentlichungen der gesponserten Projekte, Veranstaltungen und Vorträge zu finden und gleichzeitig dient diese Adresse auch als zentrales Directory für die Vielzahl von Arbeitsgruppen, Organisationen und Projekte, die sich mit dem Thema Sicherheit in der Softwareentwicklung beschäftigen (→SwA Communities).

BSIMM

Das Building Security In Maturity Model (BSIMM – gesprochen „bee simm", http://bsimm2.com) entstand aus der Zusammenarbeit mehrerer Sicherheitsspezialisten. BSIMM wurde als Studie federführend von dem namhaften Autor zahlreicher Standardwerke aus dem Bereich Informationssicherheit, speziell im Bereich der Softwareentwicklung, **Dr. Gary McGraw**, gleichzeitig CTO von **Cigital**, und der Fa. **Fortify** (kürzlich von HP übernommen), erstellt.

BSIMM wurde entwickelt, um Softwaresicherheit mess- und planbar zu machen. Nach eigenen Angaben war der Ansatz von

BSIMM die Hoffnung und der Plan, die Behandlung der Software-Sicherheit nicht mehr als „Alchemie" verstehen zu müssen, sondern als „empirische Wissenschaft" zu etablieren. Ausgehend von den Vorarbeiten zu Microsofts SDL wurden die Entwicklungsprozesse von 30 anderen Firmen (davon in der 2. Version auch neun europäische Unternehmen) mit dem SDL und untereinander verglichen. Zu den beteiligten Unternehmen gehören neben High-Tech-Firmen auch Banken und ein Telekommunikationsunternehmen.
Aus den gewonnen Erkenntnissen wurde eine Studie erstellt, mit deren Hilfe Softwareentwickler ihre Prozesse unter die Lupe nehmen können. Die BSIMM-Studie steht jetzt bereits in der zweiten Version (BSIMM2) gratis zum Download bereit und hat sich seit Erscheinen der ersten Version im Jahr 2009 laut McGraw inzwischen zum De-facto -Standard gemausert. McGraw positioniert BSIMM dabei nicht als Methode, sondern als Mess-Werkzeug, einem Zollstock gleich, an dem der eigene Reifegrad abgelesen werden kann.
BSIMM umfasst Gebiete wie das Prüfen von Sourcecode, Angriffsmodelle oder Penetrationstests. Das Interessante am Entstehen der Studie ist, dass die insgesamt 110 untersuchten Faktoren nicht vorher festgelegt wurden, sondern die Studie „live" nach und nach anhand der von den Unternehmen in der Praxis genutzten Mechanismen entstanden ist.

Etwa 6-7 Jahre nach dem Aufbau und der Einführung des SDL von Microsoft ist BSIMM entstanden und bilanziert im Ergebnis ein insgesamt positives Fazit: das Bewusstsein über die Bedeutung sicherer Software ist in allen befragten Unternehmen vorhanden und entsprechende Maßnahmen durchaus bereits aufgesetzt. Die befragten Unternehmen bestätigten auch eine zunehmende explizite Forderung der Kunden nach sauber programmierten und schwer angreifbaren Anwendungen und fordern oftmals sogar den Nachweis dafür beim Lieferanten an.

Eine gewisse Analogie, wie wir sie aus den Service-Level-Agreements (SLA) kennen: Der Dienstleister verspricht eine Leistung und eine ausgehandelte Qualität und Güte und muss selbst den Nachweis erbringen, dass diese Parameter erfüllt werden. Dieses aber kann nur dann funktionieren, wenn – wie in diesem Beispiel – seine Service-Prozesse auch intern in einer hohen Qualität und Zuverlässigkeit ablaufen und er dieses messen kann.

Das Software Security Framework (SSF)
Die Tabelle unten zeigt das zugrunde liegende Software Security Framework (SSF), um das die Ergebnisse der Studie herum aufgebaut und aggregiert wurden. Insgesamt sind 12 Verfahren/Maßnahmen benannt, die in vier Bereiche aufgeteilt sind.

The Software Security Framework (SSF)			
Governance	**Intelligence**	**SSDL Touchpoints**	**Deployment**
Strategie und Metriken	Angriffsmodelle	Architekturanalysen	Penetration Testing
Compliance und Policy	Security Features und Design	Code Review	Software Environment
Training	Standards und Requirements	Security Testing	Configuration Management and Vulnerability Management

Abbildung-2: Das BSIMM - Software Security Framework

- **Governance** fasst die Verfahren/Maßnahmen zusammen, die dabei helfen, eine Initiative für Softwaresicherheit zu organisie-ren, zu managen und zu messen. Die Ausbildung des Personals ist auch eine zentrale Governance-Maßnahme

- **Intelligence**-Verfahren/Maßnahmen führen zu einer Sammlung von Unternehmenswissen, welches verwendet wird, um wiederum Verfahren/Maßnahmen für Softwaresicherheit in der Organisation durchzuführen. Die Wissenssammlungen umfassen sowohl proaktive Sicherheitsmaßnahmen als auch organisatorisches Threat Modeling.

- **SSDL Touchpoints** befassen sich mit der Analyse und Gewährleistung von bestimmten Ergebnissen und Prozessen der Softwareentwicklung. Hierzu gehören auch alle Vorgehensweisen für die Softwaresicherheit.

- Zum **Deployment** gehören Verfahren/Maßnahmen, die mit traditionellen Organisationen für Netzwerksicherheit und Soft-warewartung interagieren. Konfiguration, Wartung und andere Aspekte der Anwendungsumgebung haben einen direkten Einfluss auf die Softwaresicherheit.

Innerhalb des Reifegradmodells werden insgesamt 110 Aktivitäten dargestellt, die jeweils einer der zwölf Verfahren/Maßnahmen zugeordnet sind. Es werden jeweils Ziele für jede Ebene der Verfahren definiert, die dann in Unterziele herunter gebrochen werden. Auf diese Weise erfolgt dann die Zuordnung zu den Aktivitäten.

Die übergeordneten Ziele vom BSIMM sind:

Sachkundige Entscheidungen im Risiko-Management
Klarheit bezüglich der adäquaten Schritte für jeden, der in die

Softwaresicherheit involviert ist Einsparung von Kosten durch standardisierte, wiederholbare Prozesse
Verbesserte Qualität des Codes
Jeder der vier Themenbereiche im SSF hat eigens zugeordnete Ziele:
Im SSF gibt es drei Verfahren/ Methoden für jeden der vier Bereiche. Letztlich werden wieder jedem der Verfahren eigene Ziele zugeordnet jeweils mit Handlungsempfehlungen versehen. Abschließend liefert das Modell eine tabellarische Darstellung als Checkliste für jeden Bereich, mit dem die eigene Organisation eingeschätzt und bewertet werden kann (siehe Originaldokument).

Bereich	Geschäftsziele
Governance	Transparenz, Rechenschaftspflicht, gegenseitige Kontrolle
Intelligence	Prüffähigkeit, Verantwortung, Standardisierung
SSDL Touchpoints	Qualitätskontrolle
Deployment	Qualitätskontrolle, Änderungsmanagement

Abbildung 3: Zuordnung von Geschäftszielen für die 4 Bereiche des SSF

Wie schon in der Einleitung dargelegt wurde, darf keine Software-Entwicklungsabteilung und kein Verantwortlicher glauben und hoffen, dass Modelle wie BSIMM ein Patentrezept oder eine Garantie für sichere Software darstellen. Allerdings darf durchaus – speziell bei BSIMM – mit hoher Wahrscheinlichkeit davon ausgegangen werden, dass die vielen „wenn" und „aber", die der Einführung einer Sicherheitsorganisation im Software-Entwicklungsprozess möglicherweise entgegenstehen, bei den in der Studie untersuchten Unternehmen auch vorherrschten. Dennoch haben diese Unternehmen „das Beste" daraus gemacht und sind „besser" geworden.

Wer sich erst heute mit diesem Thema befasst oder befassen muss, ist zwar „spät dran", kann aber durch gezielte Recherche nach „best practices" und deren individueller Anpassung auf sein Unternehmen und seine Organisation enorm profitieren.

Fortsetzung folgt...
In einem Folgeartikel wird auf die „Mutter" der Reifegradmodelle für Software-Erstellung, dem **Capability Maturity Modell (CMM)** und einige Nachfolgern, sowie dem Konkurrenzmodell zu BSIMM, dem beim OWASP eingeordneten offenen Reifegradmodell **Software Assurance Maturity Model (SAMM oder openSAMM)** eingegangen.

Ein interessanter, weil sehr pragmatischer, Ansatz findet sich bei „The Software Assurance Forum for Excellence in Code (**SAFECode**), einer Non-Profit-Organisation aus namhaften IT-Unternehmen. SAFECode hat sich das Ziel gesetzt, durch Erarbeiten und Bereitstellen von Best-Practice-Lösungen zuverlässige und sichere Software, Hardware und Services zu fördern.
Nicht zuletzt wird nach Ansätzen zur formalisierten Prüfung / Zertifizierung von Softwaresystemen im Hinblick auf Sicherheit recherchiert und darüber berichtet – es bleibt spannend!