Automationssicherheit

Security Testing – weil funktionales Testen nicht ausreicht

Software ist heutzutage allgegenwärtig. Bei vielen Geräten, etwa einer Kaffeemaschine, einer Digitaluhr, einem Herzschrittmacher oder einer Insulinspritze, wird sie kaum wahrgenommen, an anderen Geräten, etwa einem Smartphone oder Tablet, ist ihre Präsenz deutlich(er) spürbar. Software übernimmt immer mehr Aufgaben und ihr Umfang variiert stark. Die große Vielfalt macht es allerdings schwer, die Sicherheit von Software, also Safety und Security, zu gewährleisten. Von Prof. Dr. Hartmut Pohl

SAFETY BETRACHTET die Auswirkungen der IT auf die Umwelt; Security betrachtet die Sicherheit der IT vor Angriffen (aus dem Umfeld): Die Angriffssicherheit der IT. Darüber hinaus gilt: Ohne Security ist Safety nicht erreichbar.

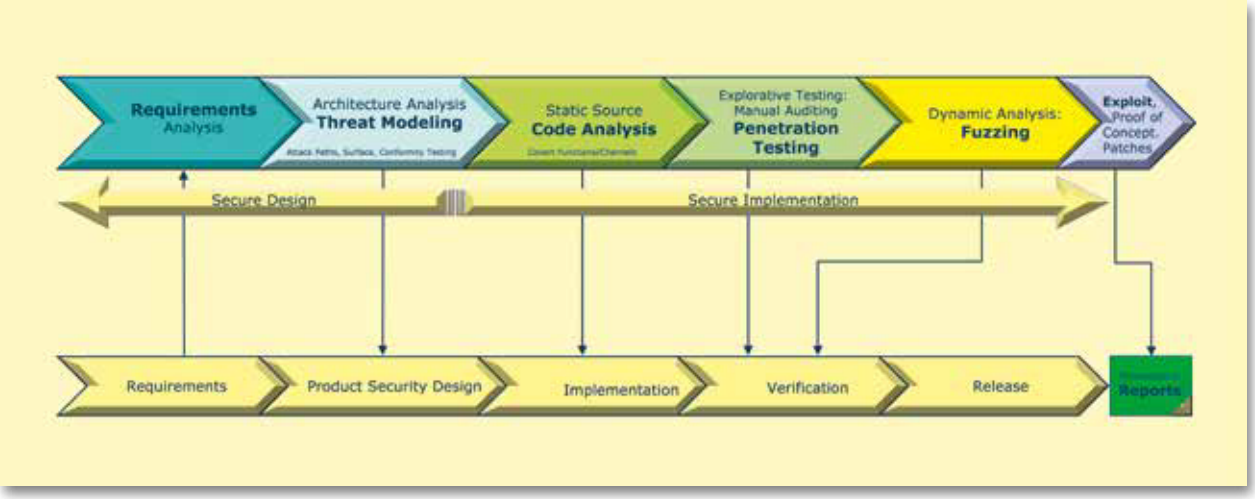
Während Software im Safety-Bereich basierend auf der weitgehend bekannten Norm IEC 61508 entwickelt wird, sind die Security-Anforderungen der entsprechenden Norm ISO 27034 weitgehend unbekannt – jedenfalls wird die Norm kaum eingesetzt.

Bedingt durch Umfang und Vielfalt der Software entstehen unter Sicherheitsaspekten Risiken, deren mögliche Folgen bisher kaum abzusehen sind – bewertet werden sie jedenfalls meist gar nicht. Einen allgemeinen Ansatz zum Management von Applikationssicherheit liefert diese 2011 von der International Standards Organization (ISO) verabschiedete ISO 27034-1. Die Norm mit der Bezeichnung „Information technology – Security techniques – Application security" bietet hierfür eine hersteller- und technologieunabhängige Grundlage. Sie definiert Konzepte, Frameworks und Prozesse, die Unternehmen helfen, Application Security in ihren Software Development Lifecycle zu integrieren. Hohe Kompatibilität und leichte Skalierbarkeit vereinfachen die Integration in bereits bestehende Strukturen und müssten eigentlich erwarten lassen, dass die Norm zügig von der Industrie adaptiert wird.

Der Einsatz herkömmlicher Verfahren zur Behebung von Fehlern und insbesondere bisher nicht erkannter Sicherheitslücken ist sehr kostenaufwändig. Und viele Sicherheitslücken werden in der Praxis nicht oder erst nach der Auslieferung der Software an die Kunden, teilweise auch von Dritten, erkannt.

Security Testing Process identifiziert Sicherheitslücken

Angesichts der Tatsache, dass Software nicht fehlerfrei erstellt werden und ein (erfolgreicher) Angriff nur unter Ausnutzung einer Sicherheitslücke erfolgen kann, ist bei der Software-Entwicklung der kostengünstige vollständige Security Testing Process entscheidend, mit dem Ziel, alle Sicherheitslücken einschließlich der bisher nicht erkannten Zero Day Vulnerabilities zu identifizieren. Dazu werden erfolgreich die folgenden Tool-gestützten Methoden eingesetzt:



Security Testing Process gemäß ISO 27034 identifiziert alle, auch bislang nicht erkannte Zero-Day-Vulnerabilities.

1. Security by Design: Entwicklung von Sicherheitsarchitekturen für Software
2. Threat Modeling: Überprüfung der Sicherheitsarchitektur (Designphase)
3. Static Source Code Analysis: Überprüfung von Implementierungsfehlern (Implementation Phase)
4. Penetration Testing: Zur Überprüfung auf bereits bekannte Sicherheitslücken (Verification Phase)
5. Dynamic Analysis – Fuzzing: Test der ausführbaren, kompilierten Datei – kein Quellcode nötig (Verification-Phase)

Bild: Marcus Kretschmar, fotolia.com

Der Einsatz dieser fünf Methoden wird umso wirkungsvoller und kostengünstiger, je früher in der Software-Entwicklung Sicherheitslücken identifiziert werden, möglichst also beginnend in der Designphase. Am teuersten wird es, wenn die Software schon ausgeliefert ist (Release-Phase) und erst dann Sicherheitslücken korrigiert werden.

Zudem lassen sich diese fünf Methoden für viele Anwendungsbereiche einsetzen: Software (Webapplications, ERM, CRM, SCM, ERP, E-Business, CIM usw.), Netzwerk-Protokolle, Embedded Systems (auch die Hardware) und Industrial Control Systems (ICS, auch proprietäre Systeme), Manufacturing Execution Systems (MES), Produktionsleitsysteme, SCADA (Leittechnik und -systeme), SPS bis zur Feldebene, Cyber Physical Systems (CPS), Industrie 4.0, Apps und Applets für smart and mobile Devices, im Cloud Computing und auch in Hardware (Firmware). Im Bereich Smart Grid und M2M wurden erfolgreich Energy-Management-Systeme (EMS) und Machine to Machine Communication sowie Smart Meter Gateways (SMGW) abgesichert.

Der Patch-Aufwand erhöht sich tatsächlich, nach einer Untersuchung des National Institute of Standards and Technology (NIST) steigt dieser von der Designphase bis zur Release-Phase auf das Hundertfache an. Aus eigenen Untersuchungen wissen wir, dass der Aufwand zur Identifizierung von Sicherheitslücken mit Threat Modeling, Static Source Code Analysis und Dynamic Analysis Fuzzing vergleichsweise kostengünstig durchgeführt werden kann, auch wenn sich die Software schon auf dem Markt befindet.

Bewertung der Sicherheitslücken nach ihrem Schweregrad

Im Security Testing Process werden die identifizierten bislang nicht erkannten Sicherheitslücken hinsichtlich ihres Schweregrads (severity) bewertet (Rating), um eine Entscheidung des Auftraggebers zu ermöglichen, ob alle identifizierten Sicherheitslücken behoben werden sollen oder ob auf das Patchen einiger verzichtet werden kann (Priorisierung).

Die entscheidenden Faktoren für die fünf Methoden sind:

- Erst der Einsatz aller fünf Methoden ermöglicht die Identifizierung tatsächlich aller Sicherheitslücken – insbesondere der bisher nicht erkannten Zero Day Vulnerabilities.
- Dynamic Analysis: Fuzzing benötigt keinen Quellcode (Blackbox-Test): Der Binärcode der Software wird während der Ausführung untersucht. Hierzu kann die Software in einer virtuellen Maschine oder auf einem anderen Testsystem ausgeführt werden.
- Im konkreten Testbetrieb hat sich herausgestellt, dass erst ein Bündel von Fuzzern eine hinreichende Menge kritischer Sicherheitslücken identifiziert: Jeder Fuzzer identifiziert erfahrungsgemäß nämlich andere Sicherheitslücken.

Ist für eine konkrete Zielsoftware ein angemessener Fuzzer nicht verfügbar, muss ein spezieller Fuzzer geschaffen werden. Für spezielle Zielsoftware müssen eventuell auch spezifische Attack Strings als Testdaten entwickelt werden.

Der Einsatz dieser Methoden führt zur Identifizierung aller Sicherheitslücken, einschließlich der noch nicht bekannten Zero Day Vulnerabilities, und macht die geprüfte Software tatsächlich angriffssicher. sg

Autor: Prof. Dr. Hartmut Pohl ist Professor für Informationssicherheit und geschäftsführender Gesellschafter der IT-Sicherheitsberatung softScheck GmbH in Köln/Sankt Augustin.