Cybersecurity für die Software von medizinischen Geräten

Anwendungssicherheit automatisiert testen

© shutterstock_Medical-1682600689

Die Digitalisierung zieht sich heute durch alle Märkte. So können in modernen medizinischen Gesundheitssystemen mit Sensoren ausgestattete Geräte die kritischen Parameter der Gesundheit des Patienten überwachen und diese Informationen an ein System in der Cloud senden, um eine automatische Analyse durchzuführen, Warnungen auszugeben oder die Medikamentenverabreichung zu ändern. Allerdings entstehen durch die Konnektivität von leistungsstarken IoT-Geräten viele Sicherheitsbedrohungen, Unternehmen müssen sich also vor Cyberangriffen wappnen. Attacken auf die Anwendungssicherheit sind die häufigste Form der externen Bedrohung. Das macht das Testen der Anwendungssicherheit (AST Application Security Testing) zu einer Schlüsselaufgabe der Sicherheitsverantwortlichen.

Was sind Anwendungssicherheitstests?

Bei ASTs kommen verschiedene Techniken zum Einsatz wie u. a. statische Sicherheitstests von Anwendungen, Penetrationstests, verschiedene Testtools. Um Qualität und Sicherheit von Softwareanwendungen zu verbessern, werden Schwachstellen und Sicherheitslücken in allen Phasen des Softwareentwicklungsprozesses identifiziert, behoben und letztlich verhindert. Der Prozess des Identifizierens und Behebens von Anwendungsschwachstellen funktioniert am besten, wenn er als Teil der funktionalen Tests integriert werden kann. Innovative AST-Tools erweitern das automatisierte Testen der Anwendungssicherheit auf den gesamten Softwareentwicklungszyklus (SDLC Software Development Life Cycle) und helfen dabei, Sicherheits- und Qualitätsprobleme aufzudecken, die Schwachstellen in Softwareanwendungen darstellen könnten.

Ihre Stärken spielen ASTs aus, wenn sie frühzeitig und oft durchgeführt werden. Dabei spielt in der modernen Softwareentwicklung die Automatisierung eine Schlüsselrolle, denn sie ermöglicht es, Softwareanwendungen schnell bereitzustellen, ohne die Sicherheit und Qualität zu beeinträchtigen.

Frühzeitig testen

Um Sicherheitsprobleme in Echtzeit anzugehen, empfiehlt sich die nahtlose Integration von Sicherheit in die Entwicklungspipeline. Das frühzeitige Erkennen von Problemen ermöglicht:

- die schnellere Bereitstellung von Software,
- das Senken der Risiken in Softwareanwendungen,
- das Reduzieren der Kosten für das Beheben von Problemen,
- die Steigerung des Sicherheitsbewusstseins der Entwickler.

Häufig testen

Die Integration von ASTs in die CI/CP-Pipeline und Toolketten gewährleistet kontinuierliche Tests, um Risiken als Folge von Codeänderungen aufzudecken. Die Automatisierung dieser Strategien ermöglicht:

- das Durchsetzen von Sicherheit und Konformität bei jeder Übertragung,
- eine höhere Transparenz in Bezug auf Anwendungssicherheit und Unternehmensrisiken,
- die Vereinfachung von Workflows zur Fehlerbehebung und Problemlösung,
- die Beschleunigung und Skalierung von Sicherheitstests, um Fehler besser zu erkennen.

Zuverlässig liefern

Die Strategie „do it early and do it often“ bietet die Gewissheit, dass Softwareanwendungen frei von bekannten Anwendungsschwachstellen sind, und Entwicklungsteams ihre Software zuverlässig liefern und einsetzen können.

Garantierte Softwaresicherheit in kürzester Zeit wird erzielt durch:

- das Einbinden von Sicherheits- und Konformitätsphasen in die Entwicklungsabläufe.
- Das Bewerten der Risiken von Softwareanwendungen in Echt-

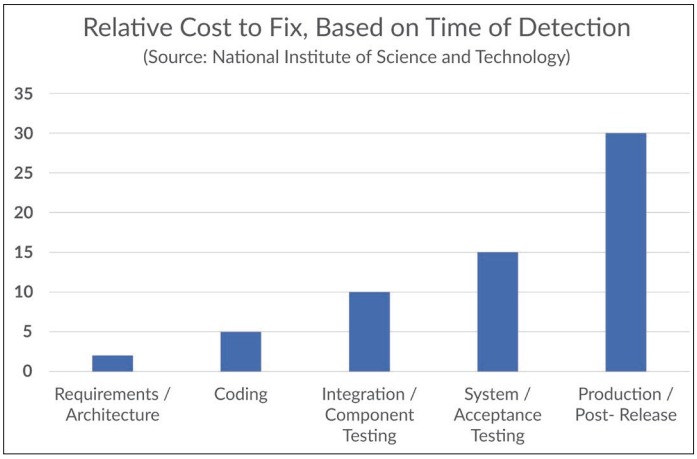

Bild 1: Der Aufwand zur Fehlerkorrektur steigt mit dem Fortschritt des Softwareentwicklungszyklus © Parasoft

Autor:
Riccardo Camacho
Parasoft
www.parasoft.com

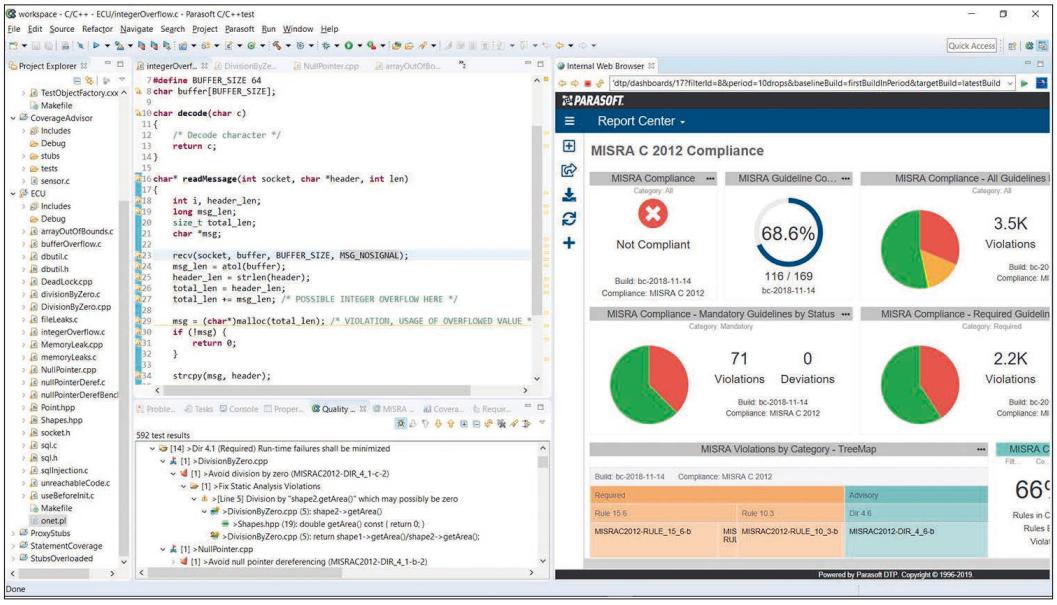

Bild 2: Moderne statische Analysetools integrieren innovative Dashboards, die sowohl Entwicklern als auch Managern aufschlussreiche Zahlen liefern © Parasoft

zeit als Grundlage für die Entscheidungsfindung.
- Das Automatisieren und Kodifizieren von Sicherheits- und Konformitätsprüfungen in Toolchains.
- Die vereinfachte Problembehebung und Triage, um sich auf das Wesentliche zu konzentrieren.

Arten von Anwendungssicherheitstests

Statische Anwendungssicherheitstests (SAST)

SAST nutzt statische Analysetechniken, um Quellcode, Bytecode und Binärdateien auf Codeverletzungen und Softwareschwächen zu untersuchen. SAST-Tools verschaffen Entwicklern Kenntnis und Feedback über die Auswirkungen ihrer Programmier- und Refactoring-Aktivitäten auf die Entstehung von Sicherheitslücken in der Software.
- Sie unterstützen bei der Durchsetzung sicherer Programmierpraktiken (CERT, CWE, OWASP).
- Sie verwenden White-Box-Tests, bei denen die Tester nicht kompilierten Code auf Fehler untersuchen.
- Sie erzwingen gute Programmierpraktiken als Präventivmaßnahme, die dazu beiträgt, die Sicherheit von der Projektkonzeption an einzubauen.

Dynamische Anwendungssicherheitstests (DAST)

Im Gegensatz dazu nutzt die DAST-Technologie Black-Box-Tests, bei denen der Code ausgeführt und dann auf Schwachstellen untersucht wird. Diese Tools können oft umfangreichere Überprüfungen durchführen, indem sie schlecht durchdachte Testfälle und unerwartete Vorfälle simulieren.

Interaktive Prüfung der Anwendungssicherheit (IAST)

IAST kombiniert sowohl DAST- als auch SAST-Werkzeuge, um eine umfassendere Liste von Sicherheitsschwachstellen zu erstellen. Diese Tools überprüfen Software dynamisch während der Laufzeit, arbeiten aber auf einem Anwendungsserver. Dadurch können sie kompilierten Code überprüfen. IAST-Tools eignen sich hervorragend für API-Tests sowie für die Überprüfung von Komponenten Dritter und des Datenflusses.

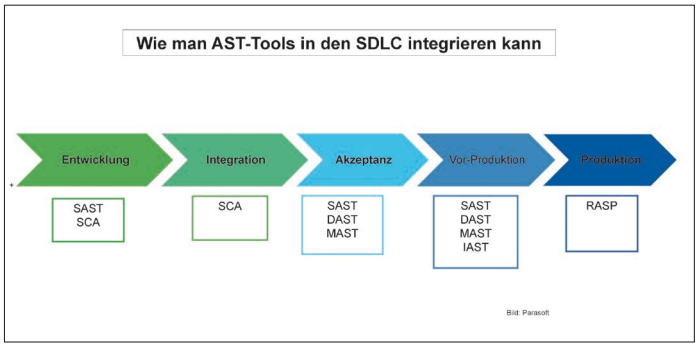

Bild 3: Möglichkeiten zur Integration von AST-Tools © Parasoft

API-Sicherheitstests

Schlecht entwickelte und durchlässige APIs sind kontraproduktiv für den Schutz des Unternehmens, des Auftrags und der Kunden. Das macht das Aufdecken eines Fehlgebrauchs und Missbrauchs von API-Funktionen entscheidend für API-Sicherheitstests. Um Sicherheitsbedrohungen aufzuspüren, die in APIs eingebettete sensible Daten preisgeben, und einen API-Angriff zu verhindern, kommen DAST und Penetrationstests zum Einsatz

Best Practices

Die Softwaresicherheit von Medizintechnik ist in den letzten Jahren zusehends in den Fokus gerückt. So konnten inzwischen Erfahrungen gesammelt und in einer Liste an Empfehlungen gebündelt werden:
- **Automatisieren** - Mithilfe von automatisierten Tools in den Entwicklungsprozessen den Lebenszyklus der Softwareentwicklung (SDLC) verbessern.
- **Überprüfen** - Komponenten und der Code von Drittanbietern oder Open Source sollten immer überprüft werden.
- **Robustheit** - Verwendung von robusten Testfällen, die bösartige Angriffe einschließen.
- **Interface-Tests** - Nicht nur Benutzeroberflächen und APIs testen, sondern auch Schnittstellen.
- **Umfassende Tests** - Mithilfe von statischer und dynamischer Analyse (IAST) die Grundlagen für umfassende Softwaretests schaffen.
- **Linksverschiebung** - Anwendung einer DevSecOps- oder „Shift Left“ (Vor-Verlegen im Entwicklungszyklus)-Strategie.
- **CI/CD-Arbeitsablauf** - Integration von AST in die CI/CD Pipeline.
- **Simulationen** – Überprüfen der Prozesse zur Risikobewältigung mit Simulationen, um zukünftige Datenschutzverletzungen zu verhindern.
- **Einblicke zum Handeln** - Geduldig sein, wenn die Teams Sicherheitsrisikodaten in umsetzbare Erkenntnisse umwandeln, die in zukünftigen Code einfließen können.

Einstieg in das Testen der Anwendungssicherheit

Es gibt viele Möglichkeiten, AST-Tools in den Softwareentwicklungszyklus von Medizingeräten einzubinden. Bild 3 zeigt die empfohlenen Tools für das Testen der Anwendungssicherheit, die in jeder Phase eingesetzt werden sollten. Ein wichtiger Teil der optimalen Nutzung dieser Tools ist das Automatisieren von Prozessen, um zeit- und kostenintensive manuelle Tests zu ersetzen. Es ist eine logische Ergänzung zur ‚Shift Left‘-Strategie der Vorverlegung von Tests in einen früheren Entwicklungszyklus, wo sich Fehler noch einfacher beheben lassen bzw. Schwachstellen in der Software leichter korrigiert werden können. Die Automatisierung unterstützt das Entwicklungsteam, indem sie die Effizienz und Produktivität steigert und zugleich Fehler reduziert. Wie man die CI/CD-Pipeline automatisiert bzw. wie dies für das eigene Team funktionieren kann, oder wie ein DevSecOps-Ansatz und Continous Testing Sicherheitsprobleme entschärfen kann, lässt sich mithilfe von Demos innovativer Lösungsanbieter feststellen. ◄