DevOps, Agile Entwicklung und Security
Passt das zusammen?

Dr. Ralf Huuck, Synopsys

Moderne Softwareprozesse sind stark von kurzen Produktzyklen, Modularität, und schnellem Markteintritt angetrieben. Dieses spiegelt sich in den sogenannten DevOps Prozessen wieder als auch in dem Trend zu zunehmend agileren Ansätzen, die eine flexible Antwort auf sich ändernde Anforderungen gibt. Auf der anderen Seite sind die Sicherheitsanforderungen insbesondere im Embedded Software Bereich stark gestiegen. Sicherheit erfordert aber gründliches Planen, eine saubere Sicherheitsarchitektur und Spezialkenntnisse, die einem ad hoc Ansatz entgegenstehen. Dieser Vortrag beleuchtet diese scheinbaren Gegensätze und die sich daraus ergebenden Konsequenzen.

Einleitung

Softwaresicherheit ist eine große Herausforderung für die Entwicklung von eingebetteten Systemen. Insbesondere die „Security" von Systemen, die mit der Außenwelt agieren, benötigt ein Expertenwissen, das oft rar ist [1]. Darüber hinaus ist die Anwendungssicherheit ursprünglich nicht im Fokus bei Design- und Implementierungsentscheidungen von eingebettetem System gewesen. Dieses wandelt sich aber in einem Maße in dem zum einem Software an größerer Prominenz gewinnt und zum anderen ein gesteigertes Sicherheitsverlangen vorliegt. Als solches wandelt sich auch die Systemindustrie zunehmend zu einer Softwareindustrie.

Veränderte Entwicklungsprozesse: Agile und DevOps

Einer der großen Trends in der Softwareindustrie ist es, Produktzyklen zu verkürzen, um schneller und flexibler am Markt zu sein. Software erlaubt es oft kontinuierlich neue Fähigkeiten als Upgrades hinzuzufügen, um das Produkt über die Softwareeigenschaften zu verbessern und weiterzuentwickeln. Das Verkürzen der Produktzyklen geht oft einher mit einem sogenannten *agilen Entwicklungsprozess*. Das bedeutet, dass in kurzer Abfolge neue Fähigkeiten definiert werden, die Architektur dafür eventuell erweitert wird und eine Implementierung erfolgt, die das bestehende Produkt erweitert und die teilweise inkrementell zu der Vorgängerversion getestet werden kann. Ziel ist es, eine kontinuierliche Produktverbesserung anbieten zu können, während man gleichzeitig flexible auf Kundenfeedback reagiert.

Der agile Entwicklungsansatz wird heutzutage um das sogenannte *DevOps* erweitert, wobei nicht nur die Softwareentwicklung agil gehandhabt wird, sondern wobei auch der Geschäftsprozess inklusive das Zusammenstellen der Software und die Auslieferung an den Kunden weitestgehend automatisiert werden.

DevOps und seine Automatisierung stellt eine zusätzliche Herausforderung an die Softwaresicherheit. Im Gegensatz zum Wasserfall-Entwicklungsmodell sind die Entwicklungszyklen oft extrem kurz, die eigentliche Implementierung ändert sich stetig und der Testprozess wird oft begleitend statt in einem separaten nachfolgenden Prozess betrieben. Dieses widerspricht den prinzipiellen Sicherheitsanforderungen, die eine stabile sichere Architektur sowie einen oft langwierigen und manuellen Testprozess verlangen, der zum Beispiel ein sicherheitsrelevantes Penetrationstesten ermöglicht.

DevOps und Anwendungssicherheit
Um den Sicherheitsherausforderungen im DevOps Prozess zu begegnen, ist es unerlässlich auch einen großen Grad an Automatisierung in den Sicherheitstestprozess miteinzubeziehen. Dafür ist es entscheidend, Softwaretools einsetzen zu können, die nicht nur automatisch ablaufen, sondern die auch direkt im Entwicklungsprozess eingesetzt werden können. Als Folge dessen verschiebt sich aber auch der Prozess des Sicherheitstestens zunehmend von separaten und speziellen Testteams hin zu Softwareentwicklern.

Im Folgenden beschreiben wir einige der Werkzeuge und Methoden, die sich zum Sicherheitstesten durch den Softwareentwickler eignen. Einige dieser Methoden sind bereits aus der Qualitätsprüfung bekannt, andere haben sich in letzter Zeit speziell für die Softwaresicherheit entwickelt.

Werkzeugklassen zum Sicherheitstesten
Die folgende Übersicht zählt exemplarisch einige der Werkzeuge auf, die vollautomatisiert benutzbar sind und somit insbesondere den DevOps Prozess unterstützen können. Wir unterscheiden folgende Werkzeugklassen:

Statische Programmanalyse
Die statische Programmanalyse hilft, Programmierfehler direkt während der Entwicklung zu entdecken und zu beheben. Diese beinhalteten Fehler wie zum Beispiel SQL Injections, Speicherüberläufe oder nicht-validierte Eingaben. Oft hilft die statische Analyse der Programmentwicklung, um Sicherheitsfehler frühzeitig zu erkennen und ihnen vorzubeugen [3]. Typischerweise läuft die statische Analyse direkt im Entwicklungsprozess.

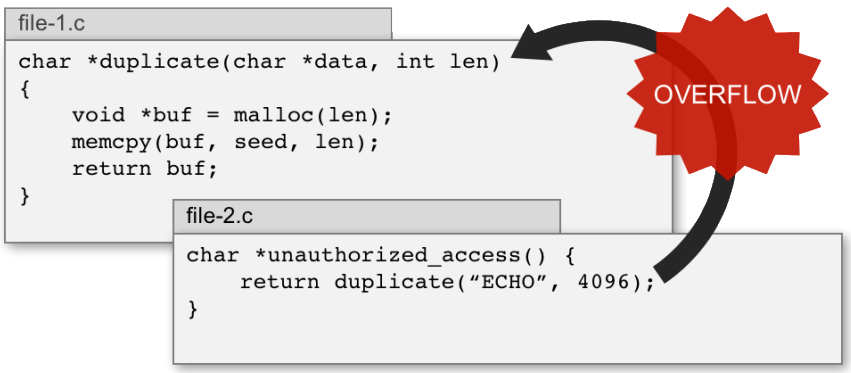

Abb. 1 Beispiel von statischer Analyse: Automatische Overflow Erkennung

Eingebettete Laufzeitmonitore
Laufzeitmonitore beobachten das Verhalten eines Systems während der Ausführung. Sie sind in der Lage, automatisch bestimmte Fehlerklassen wie verdächtige Eingaben zu entdecken. Laufzeitmonitore eignen sich sowohl zur Komplementierung funktionaler Tests als auch zur schützenden Begleitung während der Programmausführung. Darüber hinaus gibt es aktive Laufzeitmonitore, die aktiv Eingaben testen und deren Reaktionen beobachten können. Dieses kann insbesondere

für Webanwendungen hilfreich sein, um das Verhalten von möglichen Angreifern aufzudecken.

Automatisches Fuzzing von Protokollen
Fuzzing beinhaltet das Testen von Systemen mir einer großen Anzahl pseudo-zufälliger Eingaben und Daten [5]. Es wird häufig beim Penetrationstesten verwendet und kann das Verhalten eines Angreifers simulieren. Das Protokollfuzzing geht einen Schritt weiter: Hierbei wird das Hintergrundverständnis über die eigentlichen Protokolle verwendet, um automatisch gezielte Eingaben zu erstellen, die mögliche Schwachstellen der Protokollimplementierung abklopfen.

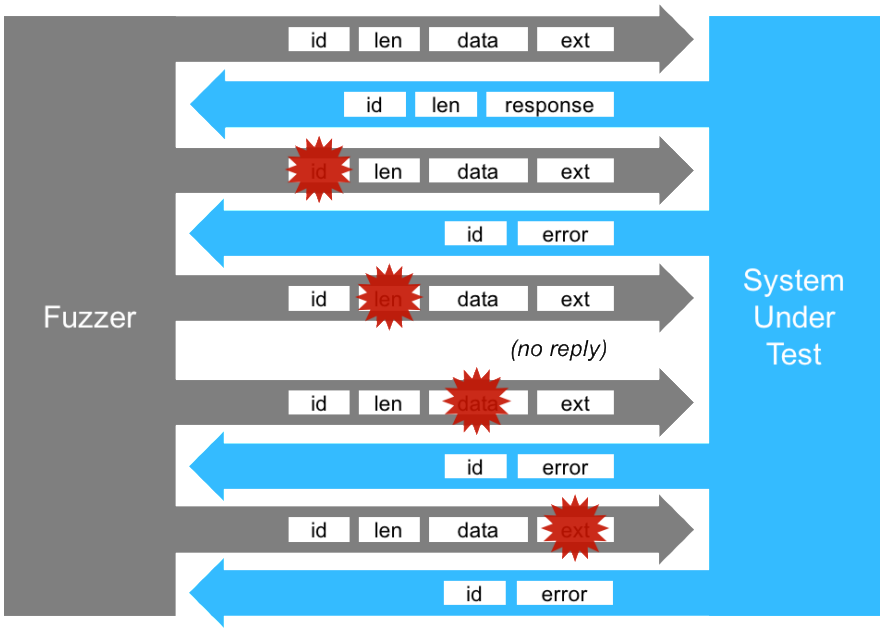

Abb. 2: Fuzzing Übersicht: Protokollein- und Ausgaben testen.

Binäranalyse von Drittanbieterkomponenten.
Eine große Herausforderung im DevOps Prozess ist es, Software von Drittanbietern einzubinden und zu testen. Dieses können sowohl Open Source Anwendung und Libraries sein als auch binäre Komponenten von Zulieferern. Um dieses Problem Herr zu werden, gibt es automatische Scanner für Source- als auch Binärcode, die Software gegen eine Liste von bekannten fehlerbehafteten Libraries und Code prüft. Diese Listen beinhalten oft CVEs und CWEs [4], die von NIST oder anderen Organisationen gepflegt werden.

Herausforderungen
Trotz dieser Reihe neuer automatischer Werkzeuge für die Verbesserung von Softwaresicherheit bleibt ein Restrisiko, das teilweise höher im DevOps Fall als im traditionellen Ansatz ist. Zum einem liegt dieses an der Fehlbarkeit und Beschränkung von Werkzeugen, beziehungsweise, dass diese Werkzeuge oft nicht alle Fälle abdecken und der Mensch weiterhin als Tester benötigt wird. Zum anderen, vernachlässigen diese Werkzeuge die allgemeine Architektur, die, falls nicht korrekt designt, ein falsches Sicherheitskonzept vorgibt. Als solches, bleibt der Mensch weiterhin unerlässlich, um sichere Software zu designen und zu testen.

Zusammenfassung
Die hier vorgestellten Prozesse und Werkzeuge ermöglichen es die allgemeine Softwaresicherheit von eingebetteten Systemen teils drastisch zu erhöhen. Die Möglichkeit diese Werkzeuge vollautomatisch ablaufen zu lassen, eröffnet die Option, diese Werkzeuge direkt in den DevOps Prozess mit einzubinden. Während diese Werkzeuge nicht den kompletten Sicherheitsprozess abdecken können, ermöglichen sie es dennoch, den Menschen gezielter und effektiver einzusetzen.

References
[1] John Viega & Gary McGraw. Building Secure Software. Addison-Wesley - ISBN 0-201-72152-X.
[2] James D. Brown. Mythbusting: DevOps and Security. Wired Magazine, 2013. https://www.wired.com/insights/2013/10/mythbusting-devops-and-security/
[3] Ralf Huuck, Ansgar Fehnker, and Rodiger Wolf. Model Checking Dataflow for Malicious Input. Proceedings of the 6th Workshop on Embedded Systems Security Taipei, Taiwan, Oct 2011. ACM, Article 4, 10 pages, ISBN: 978-1-4503-0819-9.
[4] Common Weakness Enumeration. http://cwe.mitre.org/
[5] Barton Miller. In Ari Takanen, Jared DeMott and Charlie Miller, Fuzzing for Software Security Testing and Quality Assurance, 2008, ISBN 978-1-59693-214-2

Autor
Dr. Ralf Huuck ist ein Director und Senior Architect bei Synopsys, Australien. Dr. Huuck arbeitet im Werkzeugbereich der Synopsys Software Integrity Group und unterstützt die Werkzeugentwicklung, um standardkonforme Produkte sicher und schnell zu an den Markt zu bringen. Zuvor war Dr. Huuck Geschäftsführer von Red Lizard Software, Sydney, und langjähriger Forschungs- und Entwicklungsleiter am Forschungsinstitut, NICTA. Zeitgleich ist Dr. Huuck Adjunct Associate Professor an der UNSW, Australien, und ist im Bereich Softwaresicherheit tätig.

Kontakt
Internet: www.synopsys.com.software
Email: ralf.huuck@synopsys.com