Open Source Sicherheit: Zeitbomben in meiner Software?

Wie ich Open Source (und andere) Software richtig manage

Dr. Ralf Huuck, Synopsys

Aktuelle Untersuchungen haben gezeigt, dass in weit über 90% aller neuen Softwareprojekte Open Source verwendet wird [6]. Dieses ist nur natürlich, wenn man bedenkt, wie weitverbreitet Open Source Standardpakete und Werkzeuge sind. Die Frage, die sich dabei stellt ist, wie kann man sicherstellen, dass diese Open Source Komponenten den eigenen Qualitäts-, Lizenz- und Sicherheitsansprüchen gerecht werden? Dieser Artikel beleuchtet einige der Risiken beim Einbetten von Drittanbieterkomponenten und stellt die Ergebnisses einer weltweiten Untersuchung zu Sicherheitslücken in Open Source Komponenten vor. Daraus abgeleitet wird erläutert, wie automatische Software Scanning Lösungen verwendet werden können, um diese Sicherheitslücken und Risiken im Entwicklungsprozess automatisch zu erkennen und eine Auslieferung im Produkt zu verhindern.

Einleitung

Open Source Software ist ein wichtiger Bestandteil vieler Produkte. Dabei ist die Sicherheit von Open Source Software generell vergleichbar mit selber entwickelter Software. Das Prinzip der „many eyeballs“ hilft, entsprechende Anforderungsstandards zu erreichen. Umgekehrt bedeutet dieses aber, dass Angreifer ebenfalls die verwendeten Open Source Pakete untersuchen können, um eventuelle Schwachstellen auszunutzen. Kritisch ist, dass Sicherheitslücken oftmals erst nach längerer Zeit bekannt werden, wenn die Pakete schon im Produkt verbaut sind.

Softwaresicherheit spielt eine zunehmend größere Herausforderung bei der Entwicklung von eingebetteten Systemen. Dieses ist umso mehr der Fall, je größer Softwareentwicklungsprojekte sind und je schneller sie sich fortentwickeln. Aktuelle Beispiele finden sich in Fahrerassistenzsysteme und Industrie 4.0 Applikationen, wo die Software Millionen Zeilen von Code umfassen kann und gleichzeitig sicherheitskritische Aufgaben erfüllen muss.

Ein Beispiel für die Auswirkungen von Sicherheitslücken durch Drittanbieter ist das MIRAI Botnet [1]. Dieses Botnet umfasst in seinen verschiedenen Inkarnationen über 130.000 eingebettete Systeme, zumeist internetfähige Überwachungskameras, die durch eine gemeinsame Sicherheitslücke in einem zugelieferten Bauteil von Hackern übernommen wurden. Obwohl die Überwachungskameras von verschiedenen Herstellern vertrieben werden, basierten sie auf Lieferketten mit demselben Board und derselben Treibersoftware mit derselben Sicherheitslücke.

Im Folgenden präsentieren wird einige Ergebnisses zur Untersuchung von Risiken von eingebetteten Softwarekomponenten. Wir gehen den Fragen nach: wie häufig haben Softwarekomponenten Sicherheitslücken und wie kritisch sind diese Lücken? Darüber hinaus stellen wir einige Lösungen vor, um diese Sicherheitslücken automatisch im Entwicklungsprozess zu erkennen und zu vermeiden.

Wachsende Zahl Open Source Nutzung und bekannten Schwachstellen
Die Nutzung von Open Source ist heutzutage der Normalfall und generell ist Qualität vergleichbar mit intern entwickelter Software. Allerdings, werden oft keine speziellen Qualitätskontrollen und Reviews von Open Source Paketen durchgeführt, die von extern eingebunden sind. Oft wird dem „many eyeballs“ Prinzip vertraut. Dieses ist insbesondere deshalb erstaunlich, weil bis zu 96% aller neuer Softwareprojekte Open Source verwendet wird [6].

Schwachstellen in Open Source Software sind oftmals gut dokumentiert, sobald diese bekannt werden. Patches stehen häufig bereit, werden aber oft verzögert eingepflegt. Die US Behörde NIST pflegt die primäre Datenbank an Sicherheitslücken von Open Source Software. Diese National Vulnerability Database (NVD) [3] umfasst etwa 90.000 Einträge, die bekannte Sicherheitslücken (CVEs) dokumentieren und ihnen eine sicherheitskritische Metrik gemäß des Common Vulnerability Scoring System (CVSS) zuweist [4].

Abbildung 1 zeigt die neu hinzugekommenen Sicherheitslücken per Jahr. Dieses bedeutet nicht, das Open Source Software schlechter geworden ist, sondern vor allem, dass ihre Anzahl gestiegen ist und dass sie zunehmend nach Schwachstellen untersucht wird.

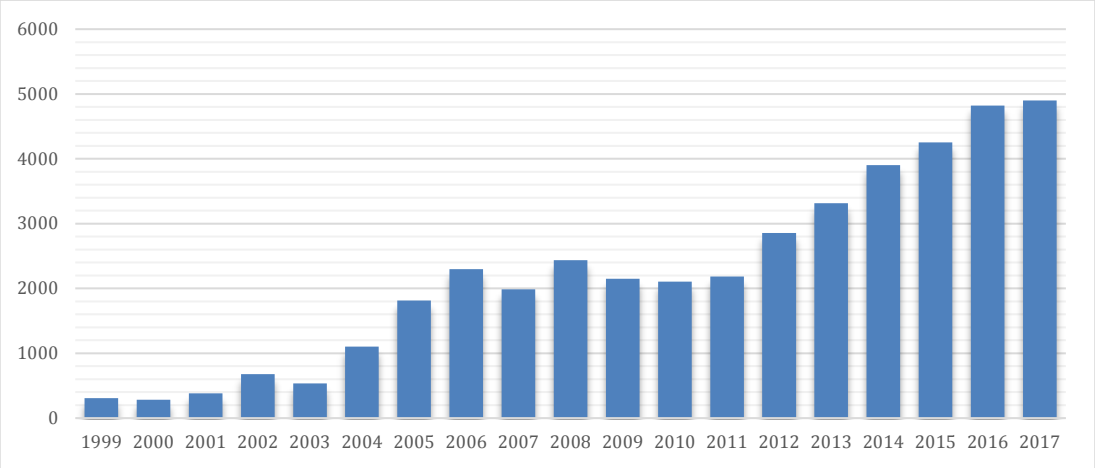

Abb. 1 Neue Open Source Schwachstellen pro Jahr [NVD/Black Duck]

Ergebnisses zu Sicherheitslücken in Open Source Komponenten.
Synopsys publiziert regelmäßig Untersuchungen zu Sicherheitslücken in Open Source Komponenten. Dabei werden zusammenfassende Ergebnisses von Software, die auf der Protecode Plattform hochgeladen wurde, veröffentlicht [2]. Protecode ist eine Analysesoftware, die Binärdateien auf bestehen Open Source Komponenten untersucht, die Versionsnummern der Open Source Komponenten ermittelt, und diese Komponenten mit bestehenden NVD Einträgen abgleicht.

Resultate für eingebettete Open Source Komponenten.
Die Studie von 2016/2017 umfasst in etwa 130,000 Uploads zur Protecode Plattform. Dabei konnten über 16.000 verschiedene Komponenten und Versionen identifiziert werden. Abbildung 1 zeigt eine Untergliederung der häufigsten identifizierten

Komponenten nach Aufgaben- und Nutzungsbereich. Dabei sind etwa zwei Drittel aller Komponenten Utilities für Windows- und Linux-Tools, Netzwerkprotokolle wie SLL und http, und Medienbibliotheken für jpg, png oder XML.

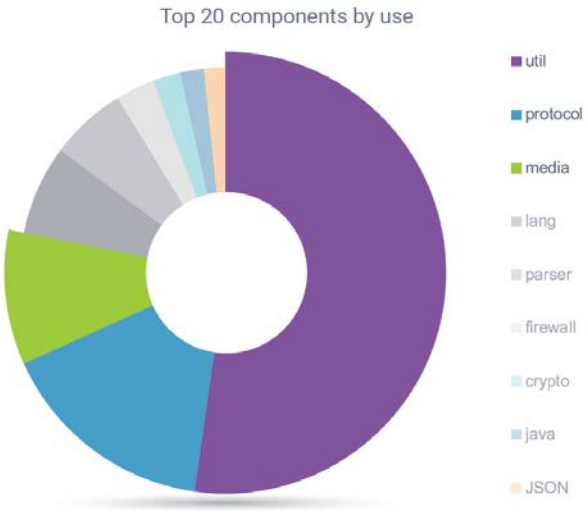

Abb. 2 Gliederung von Komponenten nach Nutzungsbereich

Diese Komponenten werden typischerweise als Open Source verwendet, da sie Standardfunktionen vereinfachen und in der Regel nicht neu implementiert werden müssen. Fälschlicherweise wird davon ausgegangen, dass diese Komponenten sicher sind oder keinen sicherheitskritischen Einfluss haben.

Es zeigt sich aber, dass aus den über 16.000 verschiedene Komponenten etwa 9.000 Sicherheitslücken (CVEs) identifizieren lassen. Das bedeutet, dass eine Großzahl der Komponenten nicht sicher ist. Darüber hinaus sind die gefundenen Sicherheitslücken nicht neu, wie Abbildung 3 zeigt:

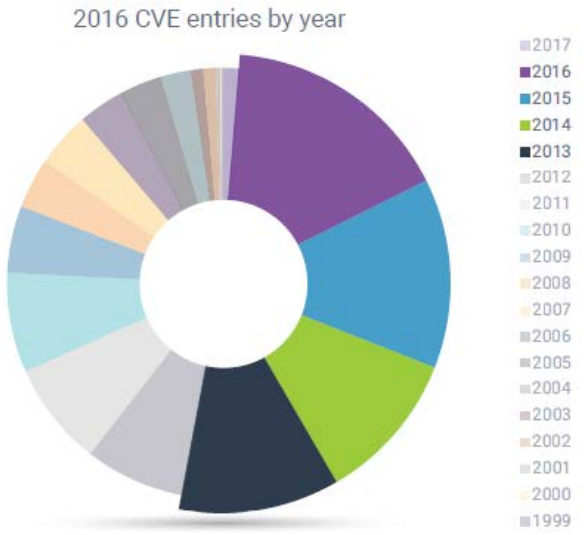

Abb. 3 Zeitraum seit Bekanntgabe der endeckten Sicherheitslücken

Dieses bedeutet, dass etwa 50% aller Sicherheitslücken vier Jahre alt oder älter sind.

In den meisten Fällen gibt es eine neuere und sicherere Version, die zur Verfügung steht, aber nicht benutzt wurde. Diese zeitliche Diskrepanz hängt auch damit zusammen, dass es oft einige Zeit dauert, bis Sicherheitslücken entdeckt werden. Das heißt, dass Software die heute als sicher gilt, morgen neuere Erkenntnisse haben kann. Dieses ist für Hersteller schwer zu kontrollieren und zu korrigieren.

Automische Einbindung von Open Source Scanning in den SDLC
Es ist unrealistisch, auf Komponenten von Open Source Quellen zu verzichten. Dieses ist ein nützlicher Bestandteil, um Produkte kostengünstig und relativ schnell herzustellen. Darüber hinaus ist es keineswegs erwiesen, dass Open Source Komponenten schlechter als eigenentwickelte Software ist. In der Tat, das Umgekehrte ist häufig der Fall.

Dennoch ist es nicht empfehlenswert, Komponenten von Drittanbietern ohne vorherige Prüfungen einzubauen. Glücklicherweise gibt es auf dem Markt heutzutage Softwarelösungen wie Protecode, Sonatype oder Black Duck, die diese Überprüfungen automatisch vornehmen können [5].

Idealerweise werden regelmäßige Scans durchgeführt, die diese Überprüfungen vornehmen. Dabei ist es hilfreich, ein Risikomodell zu erstellen, und den Software Release daran zu koppeln. Beispielhaft ist dieses in Abbildung 4 dargestellt.

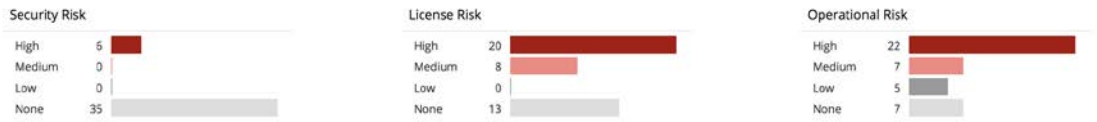

Abb. 4 Risikoklassifizierung für Open Source Software

Das Sicherheitsrisiko bildet die gefundenen Schwachstellen in der überprüften Software ab, das Lizenzrisiko die Verwendung von unangebrachten Lizenzen und das operationelle Risiko besteht aus einer Kombination aus Sicherheit, Häufigkeit der verwendeten Komponenten und der Projektaktivität.

Um diese Informationen immer zeitnah breit zu halten, lassen sich diese Softwarelösungen auch direkt in den Entwicklungsprozess automatisiert eingebunden werden. Das heißt, es lässt sich zum Beispiel ein DevOps Jenkins Prozess starten, der bei jeder Produkterstellung eine automatisierte Analyse fährt, die Open Source Komponenten entdecken und mit ihren bekannten Sicherheitslücken abgleicht. Diese Ergebnisse können dann zeitnah den Softwareentwicklungsteams und den Qualitätsteams zur Verfügung gestellt werden.

Zusammenfassung und Ausblick
Die hier vorgestellten Erkenntnisse zeigen, dass Open Source weitverbreitet ist, dass sich aber auch routinemäßig Sicherheitslücken in diesen Komponenten finden lassen. Dennoch sind Open Source Pakete für die moderne Softwareentwicklung unerlässlich. Wichtig ist, dass die Komponenten nicht mit blindem Vertrauen hingekommen werden, sondern man sie auf ihre Sicherheitsaspekte kontinuierlich überprüft. Softwarelösungen im Bereich der Open Source Scanning Analysis ermöglichen dieses heutzutage effizient und kostengünstig.

Kurzfristig sind einmalige Scans hilfreich, um sich ein Bild von den eigenen Projekten und Codebasen zu erstellen. Mittelfristig sollte aber dieser Scanprozess in den Entwicklungszyklus miteingebaut werden, um eine zeitnahe Risikoabschätzung durchführen zu können. Dieses ermöglicht dann, automatische Akzeptanzschwellen zu definieren, die zum Beispiel den CI/CD Prozess abbrechen oder automatische Emails generieren. Zum anderen lassen sich häufig Verknüpfungen zwischen vorhandenen Produkten mit ihren Open Source Komponenten und neuen Forschungsergebnissen erstellen. Dieses Bedeutet, dass automatisch ein Alarm generiert wird, falls eine neue Schwachstelle in einer verbauten Komponente bekannt wird.

Darüber hinaus ist es sinnvoll, eine Update-Strategie zu entwickeln. Welches Risiko gehe ich ein, bevor ich eine Komponente update oder patche? Wie pflege ich diesen Prozess in meinem Software Life Cycle? Wer ist der Verantwortliche und welche Sicherheitskultur lebe ich? Dieses sind oft Fragen, die jenseits der Entwicklungsteams bestehen und gruppenübergreifend angegangen werden müssen.

References

[1] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou and Jeffrey Voas. DDoS in the IoT: Mirai and Other Botnets. IEEE Computer, Volume 50/7, 2017.
[2] Synopsys Software Integrity Group. The State of Software Composition 2017. https://www.synopsys.com/software-integrity/resources/analyst-reports/state-of-software-composition-2017.html
[3] Harold Booth, Doug Rike, Gregory A. Witte. The National Vulnerability Database (NVD): Overview. ITL Bulletin, December 2013.
[4] Peter Mell, Karen Scarfone, and Sasha Romanosky. 2006. Common Vulnerability Scoring System. IEEE Security and Privacy 4, 6 (November 2006), 85-89.
[5] Millar S. Vulnerability Detection in Open Source Software: The Cure and the Cause. Queen's University Belfast, 2017.
[6] Synopsys Center for Open Source Research and Innovation. Report: 2018 Open Source and Security Risk Analysis.

Autor

Dr. Ralf Huuck ist ein Technischer Direktor 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. Zeitglich ist Dr. Huuck Adjunct Associate Professor an der UNSW, Australien, und ist im Bereich Softwaresicherheit tätig.

Kontakt

Internet: http://www.synopsys.com/software
Email: ralf.huuck@synopsys.com