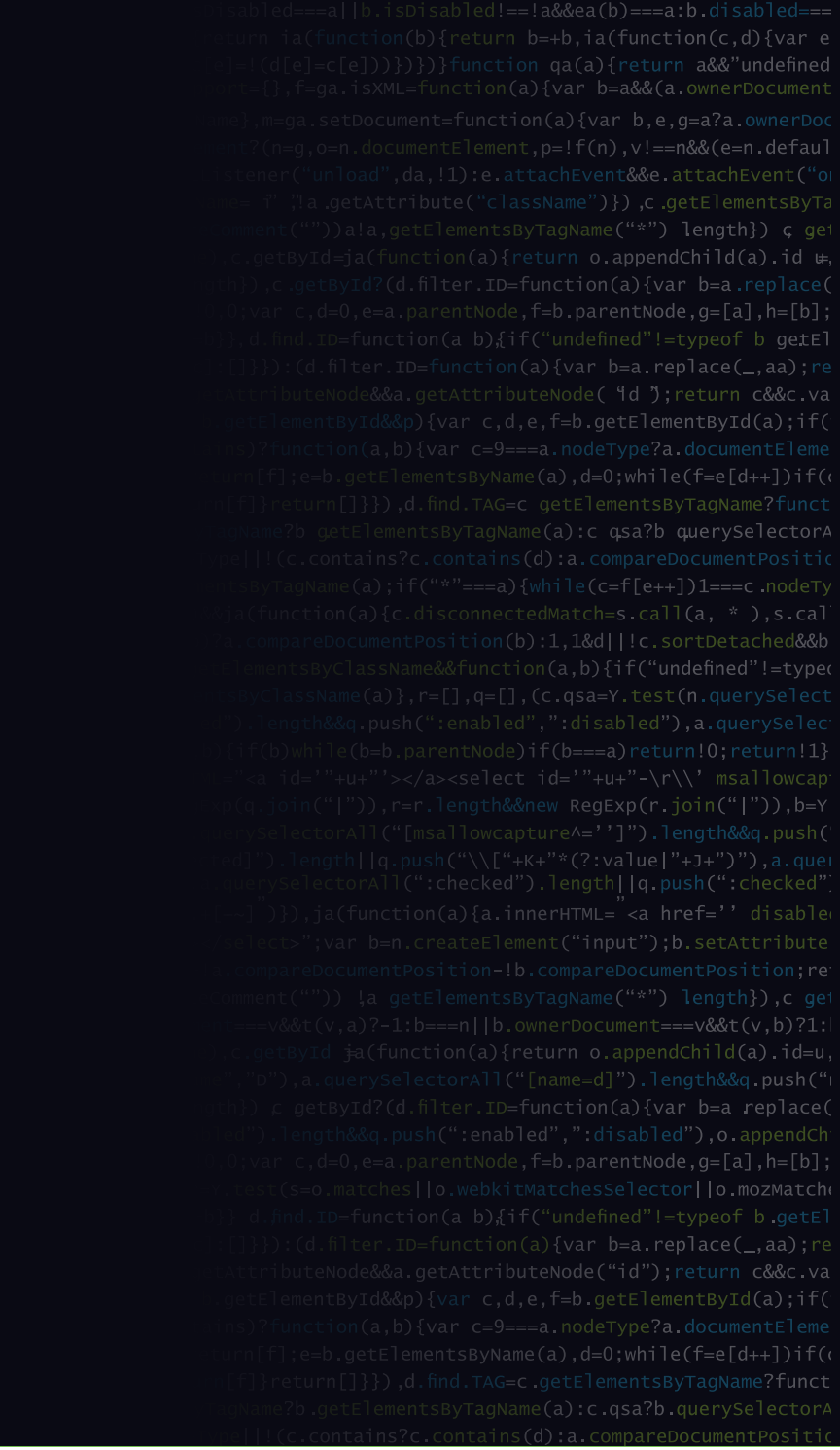
Checkmarx

Software Composition Analyse

Der ultimative Checkmarx Guide zur sicheren Open-Source-Nutzung

Mit Open Source stehen Ihnen alle Wege offen. Reisen Sie sicher

Start



Um potenziell gefährliche Schwachstellen in Software zu identifizieren, setzen entwickelnde Unternehmen in der Regel auf Lösungen für das statische, dynamische und interaktive Application Security Testing (AST), mit denen sie eigenentwickelten und kompilierten Code testen.

In vielen Szenarien liefern diese Lösungen auch wirklich überzeugende Ergebnisse. Sie wurden aber nicht entwickelt, um die Open-Source-Komponenten in Ihrer Software zu analysieren.

Dafür bedarf es einer anderen Lösung.
Software Composition Analyse.

Inhalt

+ Einführung

Die Software Composition Analyse (SCA) dient zur Erkennung und Identifizierung von Open-Source- und Third-Party-Komponenten in Anwendungen. Darüber hinaus liefert sie detallierte Risikoanalysen zu sicherheitsrelevanten Schwachstellen, Lizenzkonflikten und veralteten Libraries, die mit diesen Komponenten einhergehen.

Open-Source-Software hat die Anwendungsentwicklung nachhaltig beschleunigt und die Entwicklungszyklen deutlich verkürzt. Sie ist allgegenwärtig: Viele Analysten gehen davon aus, dass Open Source schon heute im Durchschnitt 80 Prozent der Codebasis ausmacht.

Die Nutzung von Open-Source-Komponenten geht aber auch mit Risiken einher, die es zu identifizieren, zu priorisieren und zu adressieren gilt:

Sicherheitsrelevante Schwachstellen können sensible Daten des Unternehmens angreifbar machen

Komplexe Lizenzierungsanforderungen können eine Gefahr für die immateriellen Werte des Unternehmens darstellen

Veraltete Open-Source-Libraries erweisen sich mitunter als äußerst pflege- und wartungsintensiv

Unternehmen benötigen daher einen tiefen Einblick in die Schwachstellen ihrer Open-Source-Software, einschließlich detaillierter Analysen des Gefahrenpotenzials und handlungsrelevanter Tipps zur Behebung. Genau das können moderne Lösungen für die Software Composition Analyse leisten.

KAPITEL 1

+ Open-Source-Software verstehen

+ Open-Source-Software verstehen

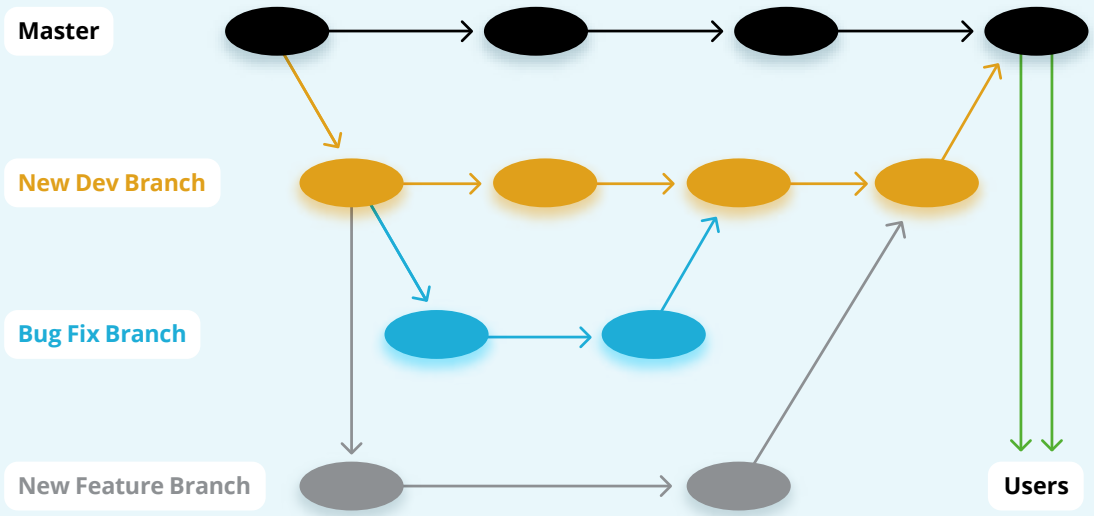
Individueller (oder eigenentwickelter) Code wird originär von einer Person oder einem Team entwickelt. Die Nutzungsrechte daran liegen dann beim Unternehmen oder bei den Autoren.

Individualsoftware wird vom Urheber oder Eigentümer des Codes gepflegt, so dass ausschließlich die Verantwortlichen den Code weiterentwickeln oder modifizieren können. Dies schließt auch auch neue Releases, Patches und Updates ein, mit denen Schwachstellen behoben werden. Individualcode kann in andere Projekte eingebettet oder als dedizierte Anwendung bereitgestellt werden.

Open-Source-Code wird ebenfalls von Entwicklern designt. Die Entwicklung wird aber in der Regel von Communities angestoßen, in denen Ideen und Beiträge geteilt werden.

Der Code oder die Software wird der Community dann in Form von Komponenten oder Projekten zur Verfügung gestellt. Da Innovationen organisch innerhalb der Community entwickelt werden, zeichnet auch die ganze Community für Updates, Patches und neue Releases verantwortlich. Wenn sich ein Projekt oder eine Komponente dynamisch weiterentwickelt, kann der Urheber des Projekts neue Lizenzierungsanforderungen oder Einschränkungen der Nutzungsrechte vorgeben.

+ Open-Source-Code entwickelt sich weiter



Open-Source-Komponenten oder -Projekte beginnen mit einem Master-Branch, der von der Open-Source-Community verändert und um neue Branches mit neuem Code erweitert wird. Ziel ist es meist, neue Features und Funktionen hinzuzufügen, Bugs zu beheben oder zu experimentieren.

Die neuen Branches werden dann in der Regel wieder in den Master-Branch eingebettet und damit zum Teil des Hauptprojekts. Sie können aber auch als separate Gabelung weitergeführt werden, etwa wenn Entwickler und Gruppen das Projekt in eine ganze neue Richtung weiterentwickeln oder für neue Use Cases anpassen.

Die Folgen der Open-Source-Dynamik

Zwei Open-Source-Komponenten mit dem gleichen Namen können relevante Unterschiede aufweisen: Der grundlegende Gedanke hinter der Komponente mag bei beiden Herstellern noch identisch sein. Doch im Detail haben vielleicht beide kleinere Anpassungen vorgenommen, um ihre konkreten Anforderungen besser abzubilden.

Weil sich jeder Branch (oft auch als Distribution oder Distro bezeichnet) dynamisch verändert, weisen die verschiedenen Versionen einer Komponente nach einiger Zeit mitunter erhebliche Unterschiede auf – auch dann, wenn sie ursprünglich auf die gleiche Komponente zurückgehen. Diese Unterschiede gehen manchmal mit zusätzlichen Pflege- und Entwicklungskosten einher oder ziehen unerwartete Security- und Compliance-Probleme nach sich.

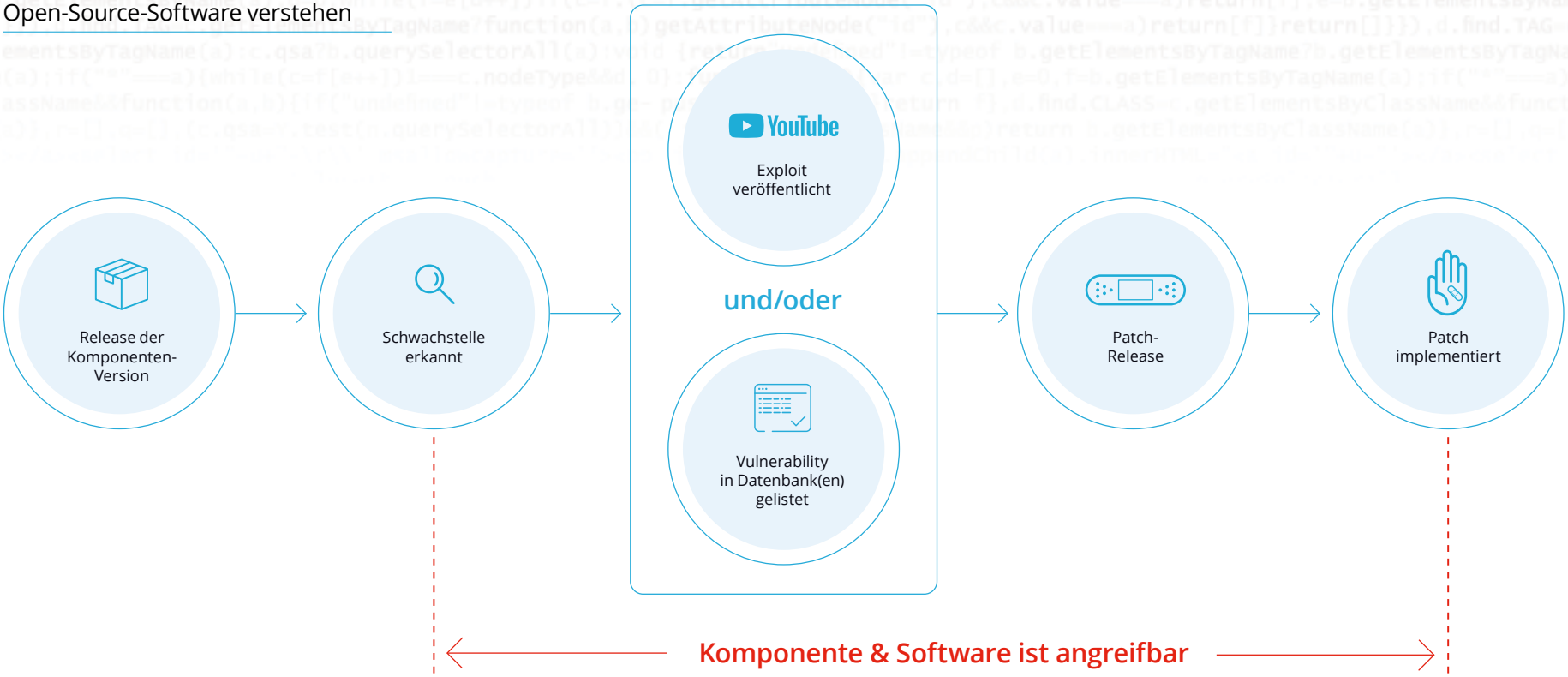
Schwachstellen in Open-Source-Code

Open Source ist heute überall, und wie Individualsoftware können auch Open-Source-Komponenten in bestimmten Szenarien angreifbar sein. Dabei ist es wichtig, zwischen unsicheren Komponenten und unsicheren Versionen einer Komponente zu unterscheiden.

Eine Komponente kann Schwachstellen aufweisen, die nur in bestimmten Versionen vorkommen. Dies hängt maßgeblich davon ab, wie die Software aufgebaut ist und wie sie sich mit der Zeit entwickelt.

- **Neue Versionen weisen oft nicht die gleichen Schwachstellen auf, die es im ursprünglichen Code oder in anderen Versionen gab.**
- **Manche neuen Versionen enthalten dafür aber Schwachstellen, die in früheren Versionen nicht enthalten waren.**
- **Eine Version ohne Schwachstellen kann im Zuge von Updates oder neuen Releases mit neuen Schwachstellen infiziert werden.**

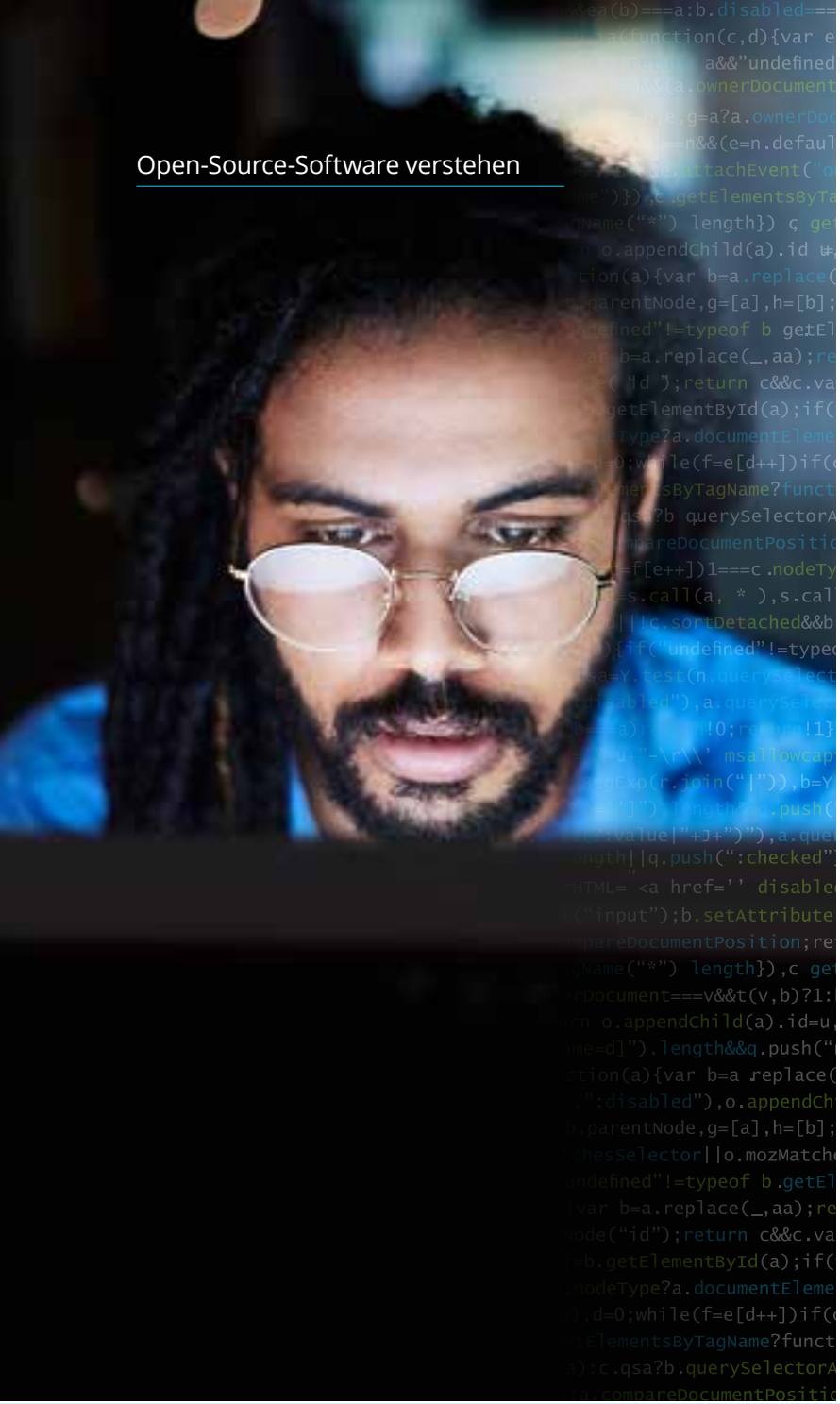
Nicht alle Open-Source-Komponenten weisen Schwachstellen auf, und nicht jede Schwachstelle kann für Angriffe missbraucht werden.



Timeline eines exemplarischen Angriffs

Angreifer müssen erst eine Schwachstelle finden und dann einen Exploit entwickeln, der diese ausnutzt und es ermöglicht, die Software zu kompromittieren. Und auch dann ist noch nicht gesagt, dass der Exploit tatsächlich ausgeführt werden kann – das hängt davon ab, wie genau die entsprechende Komponente im Code der Anwendung integriert ist.

- Nachdem eine Version einer Komponente veröffentlicht wurde, wird darin eine Schwachstelle gefunden – vielleicht vom Autor der Software, vielleicht von einem Security-Team oder von einem Angreifer.
- Manche Schwachstellen werden in einer Schwachstellendatenbank (wie der NVD) gelistet; andere werden geheim gehalten, während die, die sie gefunden haben, an einem Patch oder einem Exploit arbeiten.
- Nachdem der Patch vorliegt, vergeht einige Zeit, bevor dieser in allen Systemen implementiert wurde, die die entsprechende Komponente nutzen.
- In dieser Zeit wird manchmal ein Exploit entwickelt, der dann entweder insgeheim verwendet oder (häufiger) in einschlägigen Hacker-Communities oder auf Youtube geteilt wird, wo er für alle Angreifer verfügbar ist.



Die Zeit zwischen der Entdeckung eines Exploits und dem Einspielen des Patches ist ein gefährliches Zeitfenster, in dem Angreifer die Anwendung kompromittieren können, um Daten oder geistiges Eigentum zu stehlen oder die Performance der Anwendung zu mindern.

Es gilt also, Schwachstellen schnellstmöglich zu patchen.

+ Vom Umgang mit Vulnerability-Meldungen

Je nach Quelle und Herkunft einer Komponente wird man Sie bei Bekanntwerden einer neuen Schwachstelle automatisch informieren oder auch nicht. Handelt es sich etwa um Komponenten von Red Hat oder Apache, erhalten Sie in der Regel hilfreiche Benachrichtigungen, sobald eine Schwachstelle gefunden wird oder ein relevanter neuer Patch bereitsteht.

Für Komponenten, die von Communities entwickelt wurden, gibt es oft keine solchen proaktiven Meldungen. Damit obliegt es Ihnen, Risiken eigenverantwortlich zu identifizieren und zu beheben – mit oder ohne Unterstützung der entsprechenden Community.

Wie Sie Open Source sicher nutzen

Wenn Sie Ihren Mix aus eigenem Code und Open-Source-Komponenten zu einer robusten und sicheren Software zusammenfügen wollen, benötigen Sie die richtigen Tools, Methoden und Prozesse. Die Umgebungen, in denen Ihre Entwickler arbeiten, sind hoch komplex – und wer funktionale und sichere Software produzieren will, muss eine ganze Reihe von Details berücksichtigen.

Lizenzen, Compliance und gesetzliche Bestimmungen

Es reicht nicht, Ihren Code mit Blick auf sicherheitsrelevante Schwachstellen zu analysieren. Entwickelnde Unternehmen müssen in der Regel auch interne und externe Standards und Bestimmungen einhalten: von den SLAs des Kunden über interne Spezifizierungen bis hin zu Datenschutzvorgaben. Bei vielen Open-Source-Projekten gilt es zudem, vom Urheber vorgegebene Lizenzierungsvorgaben oder Einschränkungen der Nutzungsrechte zu berücksichtigen.

Open-Source-Projekte können unterschiedlichsten Lizenzierungsmodellen unterliegen. Grundsätzlich unterscheidet man dabei zwei Arten von Lizenzen: Copyleft-Lizenzen und freizügige Lizenzen.

- Bei Copyleft-Lizenzen unterliegt die Verteilung, Attribution und Veröffentlichung des Codes in der Regel strengen Auflagen oder Bestimmungen. Diese gelten oft auch für die Projekte, in denen die entsprechenden Komponenten eingebettet werden.

- Bei freizügigen Lizenzen sind die Vorgaben an die Distribution und Attribution der Software in der Regel minimal.

Bekannte Beispiele für Open-Source-Lizenzen sind GPL 3.0, die MIT-Lizenz und Apache 2.0. Andere Beispiele zeigen, welche hohen Freiheitsgrade die Open-Source-Autoren bei der Lizenzierung genießen: Dazu gehören die komplett offene DWTFPL-Lizenz (Do What the **** You Want To Public License) und die Beerware-Lizenz, die vorsieht, dass jeder, der eine lizenzierte Komponente verwendet, dem Autor ein Bier kauft.

Am wichtigsten ist, dass Sie sich der Lizenzvorgaben, die für Ihr Unternehmen und Ihre Anwendungen gelten, stets bewusst sind. Im nächsten Schritt können Sie dann mit den richtigen Lösungen für das Software-Security-Testing die durchgehende Einhaltung der entsprechenden Bestimmungen sicherstellen.

Anwendungstests

Wie performant, stabil und sicher die von Ihrem Unternehmen entwickelte Software ist, hängt maßgeblich von den Werkzeugen ab, die Sie Ihren Entwicklern und Ihren Security- und Ops-Teams an die Hand geben. Falls Ihre Software aus einem Mix von eigenentwickelten Komponenten und Open-Source-Code besteht, sollte Ihr Application Security Testing also in der Lage sein, potenzielle Gefahren in beiden Arten von Code zu erkennen, zu identifizieren, zu analysieren und zu beheben.

Viele Unternehmen setzen heute daher auf einen Mix aus statischem Application Security Testing (SAST), dynamischem Application Security Testing (DAST), interaktivem Application Security Testing (IAST) und Software Composition Analysen (SCA).

SAST

Analyse von Quellcode zur Identifizierung von Schwachstellen

DAST

Sogenanntes 'Black-Box'-Testing, das mithilfe von Angriffen die Funktionalität der Software prüft, aber nicht auf den Quellcode schaut

IAST

Kombinination aus SAST- und DAST-Ansätzen

SCA

Erkennt Open-Source-Code und gleicht ihn mit Verzeichnissen bekannter Schwachstellen ab, etwa der National Vulnerability Database (NVD)

SAST setzt am Quellcode an und durchsucht ihn nach Schwachstellen oder Sicherheitslücken, die Angreifer ausnutzen könnten. Diese Analyse kann je nach Umfang der Codebasis relativ viel Zeit in Anspruch nehmen und sehr viele Resultate liefern, die es im nächsten Schritt zu validieren gilt. Sämtliche aufgedeckten Schwachstellen müssen behoben und die entsprechenden Codeteile neu geschrieben werden. Dies kann viel Zeit in Anspruch nehmen und umfangreiche Ressourcen binden.

SCA untersucht den Quellcode nicht, durchsucht ihn aber nach Open-Source-Komponenten. Die Lösung muss dafür in der Lage sein, Open-Source-Code in der Software zu lokalisieren und mit einer Datenbank bekannter Schwachstellen abzugleichen. Potenziell gefährliche Komponenten, die in der Software gefunden werden, müssen gepatcht oder ersetzt werden.

Anwendungstests sind Teil des Entwicklungsprozesses

Analysieren Sie Ihre Entwicklungsumgebung, Ihre CI-/CD-Pipeline, Ihren SDLC und Ihre DevOps-Prozesse und prüfen Sie, ob Sie SAST, DAST, IAST oder SCA integriert haben. Wichtig ist, dass Sie nicht erst im Rahmen Ihrer Security-Tests nach unsicheren Open-Source-Komponenten suchen. Binden Sie die Lösungen in Ihre Prozesse ein, statt sie im Nachhinein anzuflanschen.

Wenn Sie mit Open-Source-Komponenten arbeiten, müssen Sie in der Lage sein, die einzelnen Bestandteile Ihrer Software zu analysieren, um sicherzustellen, dass diese sicher und korrekt lizenziert sind. Eine Lösung für die Software Composition Analyse (SCA) ist dafür unverzichtbar.

KAPITEL 2

+ Software Composition Analyse (SCA) verstehen

+ Software Composition Analyse (SCA) verstehen

Der Begriff Software Composition Analyse bezeichnet Lösungen, die Software analysieren, enthaltene Open-Source-Komponenten und Third-Party-Libraries erkennen und die damit verbundenen Risiken identifizieren.

Im Fokus von SCA steht die Bewertung von zwei kritischen Arten von Risiken: sicherheitsrelevante Risiken (Open-Source-Schwachstellen) und Lizenzierungsrisiken (typischerweise Lizenzverstöße oder Lizenzkonflikte). Und mitunter gibt es noch eine dritte, schwerer eingrenzbare Kategorie: Risiken in der Zusammenarbeit mit der Community.

Die ersten SCA-Lösungen, die vor etwa zehn Jahren auf den Markt kamen, fokussierten ganz auf die Überwachung der korrekten Lizenzierung von Software und der in Hardware oder Chip-Sätzen eingebetteten Technologien. Doch mit der rasanten Zunahme von Open-Source-Code verlagert sich der Use Case immer mehr auf die Software-Security: Moderne SCA-Lösungen sind ein integraler Bestandteil des Application Security Testings (AST). Inzwischen sind sogar die ersten Lösungen verfügbar, die Daten mit SAST-Plattformen teilen und korrelieren. So erkennen Sie zuverlässig, ob die angreifbaren Komponenten tatsächlich in der kompilierten Anwendung zum Einsatz kommen – und können das Risikopotenzial viel besser einschätzen.

SCA ist ein integraler Bestandteil sicherer Software-Entwicklung.

+ Wichtige Aspekte der SCA

Wenn Sie die CI-/CD-Pipeline und den Software Development Lifecycle (SDLC) Ihres Unternehmens um eine leistungsfähige SCA-Lösung ergänzen, helfen Sie den Entwicklern und den Security- und DevOps-Teams, Open-Source-Risiken priorisiert und fokussiert zu minimieren – und zwar da, wo die Behebung am effektivesten und am günstigsten ist, lange bevor die potenziell gefährliche Software in die Produktivumgebung geht.

Eine SCA-Lösung muss in der Lage sein:

- die in der Software verwendeten Open-Source-Komponenten und Versionen zuverlässig zu lokalisieren und zu identifizieren
- bekannte Schwachstellen der Komponenten und Versionen zu erkennen und Aufschluss über die Lizenzierung zu geben
- handlungsrelevante Risikoanalysen und Tipps zur Behebung zu liefern
- die Erstellung und Durchsetzung von Policies auf der Basis der Analyse-Ergebnisse zu ermöglichen
- nahtlos mit bestehenden Tools entlang des SDLC und der CI-/CD-Pipeline integriert zu werden
- den Verantwortlichen verständliche und relevante Ergebnisse zu liefern, und zwar genau in dem Format, das sie brauchen

Einige SCA-Lösungen sind darüber hinaus in der Lage:

- festzustellen, ob für eine identifizierte Schwachstelle bereits ein Exploit existiert
- festzustellen, welche Risiken von Bugs oder Community-Aktivitäten ausgehen
- Analyseergebnisse mit anderen AST-Lösungen zu korrelieren

KAPITEL 3

+ Deep-Dive in die SCA-Technologie

Deep-Dive in die SCA-Technologie

Die Software Composition Analyse erfolgt in drei Schritten:

1

Erkennung

Die Open-Source-Erkennung dient dazu, Open-Source-Komponenten in Software zu lokalisieren. Einige Detektionsansätze liefern relativ viele False Positives und nehmen viel Zeit in Anspruch. Andere sind zuverlässiger und dauern weniger lange, erfordern dafür aber etwas mehr Vorbereitungszeit für die Konfiguration.

2

Identifizierung

Im nächsten Schritt identifiziert SCA die lokalisierten Open-Source-Komponenten anhand einer umfangreichen Open-Source-Datenbank. Ermittelt werden dabei neben den grundlegenden Versionsinformationen auch Details zur Herkunft der Version sowie eine Reihe weiterer Metadaten.

3

Risikobewertung

Aufsetzend auf diesem Output bewertet die Lösung die Security- und Lizenzierungsrisiken. Hierzu gehört auch der Abgleich mit einer oder mehreren Referenzdatenbanken von Schwachstellen und Lizenzen. Einige Hersteller mit eigenem Research-Team bieten zudem exklusive Schwachstelleninformationen an.

Verfahren zur Erkennung von Open-Source-Code

Je nach SCA-Lösung gibt es verschiedene Verfahren für die Lokalisierung von Open-Source-Code: So setzen einige Lösungen auf **Signaturscans**, bei denen für jede erkannte Open-Source-Komponente eine SHA-1-Hash-Signatur generiert wird. Diese alphanumerischen Strings sind ein unverwechselbarer Fingerabdruck der jeweiligen Komponente und lassen sich einfach und schnell mit bestehenden Signaturdatenbanken bekannter Open-Source-Komponenten abgleichen.

Andere SCA-Lösungen fokussieren auf die Files, die von Ihren Build-Tools und Package-Managern generiert werden. Mit dieser **Package-Manager-Analyse** lässt sich sogar die jeweilige Version der verwendeten Komponenten feststellen – man überprüft also, ob die Software wirklich das enthält, was der Entwickler verspricht.

Und schließlich gibt es SCA-Lösungen, die auf sogenannte **Build-Dependency-Analysen** setzen und die Software etrst nach Abschluss der Entwicklung untersuchen. Damit lassen sich alle Abhängigkeiten identifizieren, die nicht offengelegt, sondern erst im Build-Prozess in die Anwendung eingebracht wurden – und nun zu potenziell gefährlichen Schwachstellen in der Software führen.

Scanning von Signaturen (oder File-Systemen)

Der größte Vorteil des Signatur-Scannings ist die hohe Zahl der Ergebnisse, die das Verfahren liefert. Auf diese Weise erhalten Sie die womöglich umfangreichste und vollständigste Auflistung der Open-Source-Komponenten in der Code-Basis.

Das Signatur-Scanning erkennt sogar nicht deklarierte Komponenten, die in den Package-Manager-Files nicht enthalten sind, und eignet sich damit auch für Software, die ohne einen Package-Manager entwickelt wurde.

Allerdings nimmt das Scanning mitunter viel Zeit und Rechenleistung in Anspruch. Außerdem liefert es sehr viele Ergebnisse, die geprüft werden müssen und oft eine hohe Zahl von False Positives enthalten. Wenn die Zeit knapp ist und die Deadline bis zur Veröffentlichung näherrückt, ist das Verfahren manchmal also eher Teil des Problems als Teil der Lösung.

Package-Manager-Analyse

Die Erkennung von Open-Source-Komponenten mithilfe eines Package-Managers hat ebenfalls einige Vorteile: Die Ergebnisse sind in der Regel sehr zuverlässig und enthalten nur wenige False Positives. Das geringe Hintergrundrauschen macht es Ihren Entwicklern und Security-Teams zudem leicht, den Output zu prüfen und die Fehlerbehebung zu priorisieren.

Die Scans sind in der Regel auch deutlich schneller, und die Technologie lässt sich wesentlich einfacher und enger in die vorhandenen CI-/CD-Tools Ihrer Entwickler integrieren.

Analyse von Build-Dependencies

Wenn die Open-Source-Komponenten nicht vollständig deklariert wurden, oder wenn der Build ohne einen Package-Manager erstellt wurde (was bei älteren Anwendungen häufig der Fall ist), wird die Package-Manager-Analyse nicht alle Open-Source-Elemente in der untersuchten Code-Basis finden. Daher kombinieren viele Anbieter die Package-Manager-Analyse heute mit einer zusätzlichen Lösung für die Analyse von Build-Dependencies.

Dieses Verfahren dient dazu, nicht deklarierte Dependencies zu identifizieren, die während des Builds oder als Dependencies von Dependencies (so genannte transitive Dependencies) eingebracht wurden.

Scanning von Snippets

Das Scanning von Snippets funktioniert ähnlich wie das Scanning von Signaturen. Statt vollständige Open-Source-Komponenten zu untersuchen, führt die SCA-Lösung dabei aber nur einen Signatur-Scan kleiner Code-Mengen durch und gleicht die Ergebnisse mit einer Datenbank zuvor gescannter und dokumentierter Komponentensegmente ab.

Mit dem Scanning von Snippets lassen sich Lizenzvorgaben, Lizenzkonflikte oder mögliche Lizenzverstöße auch dann erkennen, wenn der Entwickler nur einen Teil des Codes aus einem größeren Bestand kopiert hat. Dies setzt allerdings voraus, dass die Ergebnisse des Snippet-Scans mit einer ursprünglichen Open-Source-Komponente abgeglichen werden.

Das Scanning von Snippets nimmt natürlich viel Zeit und Rechenleistung in Anspruch, und liefert mitunter nur unscharfe Ergebnisse mit vielen ungenauen Matches und vielen False Positives. Die Resultate eines kleinen Snippets eignen sich auch nicht, um Schwachstellen zu identifizieren. Sie sind mitunter aber ein wertvolles Werkzeug, um eine korrekte Lizenzierung zu gewährleisten.

+ Identifizierung von Komponenten

Nachdem die Open-Source-Komponenten mit einer oder mehreren dieser Methoden lokalisiert wurden, müssen sie genau bestimmt werden. Dafür werden die Metadaten meist mit einer vom Anbieter gepflegten Datenbank bekannter Open-Source-Komponenten abgeglichen. Diese Datenbanken bündeln in der Regel Informationen aus unterschiedlichen Code-Repositories und Quellen wie GitHub, Maven Central und vielen mehr.

Wenn die lokalisierte Komponente in einer der Datenbanken gefunden wird, zeigt die SCA-Lösung die entsprechenden Informationen an. An diesem Punkt ist die Gefahr von False Positives am größten, und hier macht es sich am deutlichesten positiv bemerkbar, wenn Sie die richtige Erkennungstechnologie gewählt haben.

+ Risikobewertung

Nachdem die Open-Source-Komponenten erkannt und identifiziert wurden, muss die SCA-Lösung bewerten, welche Risiken mit deren Einsatz einhergehen. Nur so können Sie fundiert entscheiden, welche Bereiche Sie mit höchster Priorität angehen sollten, um die Gefahr für Ihr Unternehmen nachhaltig zu minimieren.

Dafür gleicht die Lösung die identifizierten Versionen der Komponenten zunächst mit einer Datenbank bekannter Schwachstellen und Lizenzen ab. Eventuelle Security- und Lizenzierungsisiken werden dann in der Analyse-Oberfläche der SCA-Lösung ausgegeben und den analysierten Komponenten zugeordnet.

Die Bewertung von Security-Risiken ist oft standardisiert (etwa nach CVSS2.0 oder CVSS3.0), aber diese Standardisierung ist nicht immer einheitlich. So stufen die verschiedenen SCA-Anbieter das Gefährdungspotenzial und die Priorität mitunter unterschiedlich ein. Wenn die Bewertung nicht standardisiert erfolgt, lässt sich diese in der Regel anhand der Projektkriterien risikospezifisch anpassen. Diese Funktionalität wird nicht von allen SCA-Lösungen unterstützt, ist aber wichtig, um zu verhindern, dass die relativen Risikoprofile verschiedener Projekte miteinander verglichen werden.

Lizenzierungsrisiken

Die Bewertung sicherheitsrelevanter Risiken ist in den meisten Unternehmen eindeutig in der Security-Policy geregelt. Lizenzierungsrisiken hingegen hängen maßgeblich vom Kontext und vom Einsatz-Szenario der Anwendung ab.

- Handelt es sich um eine interne Anwendung auf unseren eigenen Servern, die weder öffentlich ist noch kommerziell vermarktet wird?
- Ist die Anwendung für externe Anwender zugänglich?
- Wird die Anwendung kommerziell vermarktet?

Andere in der Anwendung genutzte Komponenten oder Lizenzen können sich ebenfalls auf die Lizenzierungsrisiken auswirken. Man spricht hier von Lizenzkonflikten: Wenn eine Anwendung etwa Open-Source-Komponenten mit Copyleft- und mit freizügigen Lizenzen enthält, kann dies mit Blick auf die Auflagen und die Attribution zu komplizierten Szenarien führen. All dies wirkt sich auf die Bewertung des Lizenzierungsrisikos aus.

Auch wenn es für die Bewertung des Lizenzierungsrisikos keine festen Standards gibt, ist man sich aber weitgehend einig, dass Lizenzen, die Geld kosten, die die Nutzung einschränken oder die Weitergabe von Nutzungsrechten an der Codebasis vorsehen, mit einem höheren Risikopotenzial einhergehen.

Mit Blick auf die SCA ist das Lizenzrisiko vor allem für die Hersteller von Embedded-Systems und Chipsätzen relevant. Hierzu gehören Use Cases im Bereich IoT sowie bei Tier-1- und Tier-2-Zulieferern der Automobilindustrie, bei Systemintegratoren und bei anderen Organisationen, die sich schwer damit tun, auf ihre Software (oder auf die Geräte, auf denen sie läuft) zuzugreifen, um sie zu ersetzen oder zu aktualisieren. In diesen Branchen kann der Verlust von Nutzungsrechten aufgrund von Lizenzkonflikten oder Lizenzverstößen verheerende Folgen haben.

KAPITEL 4

+ Worauf Sie bei der Auswahl einer Lösung achten sollten

Worauf Sie bei der Auswahl einer SCA-Lösung achten sollten

Suchen Sie eine zuverlässige Lösung mit wenigen False Positives.

Umfangreiche Ergebnisse zu erhalten, ist gut – aber nur, wenn Sie auch die Zeit haben, alle zu überprüfen. Denken Sie auch daran, dass die Risikobewertungen nicht herstellerübergreifend standardisiert sind und Gefahrenpotenziale und Priorisierungen abweichen können.

Favorisieren Sie Anbieter mit eigenem Security-Research-Team.

Stellen Sie sicher, dass der Anbieter proaktiv nach Zero-Day- und Nicht-öffentlichen Schwachstellen sucht und ein eigenes Schwachstellenverzeichnis pflegt.

Ihr Anbieter sollte eine Datenbank aller bekannten Open-Source-Schwachstellen bereitstellen.

Dazu gehören auch konkrete Hinweise zur Behebung der Schwachstellen.

Die Lösung muss die Anforderungen Ihrer Security-, Legal- und Entwicklungsteams abbilden.

Alle drei müssen Policies auf der Basis der Analyseergebnisse konfigurieren und durchsetzen können.

Priorisieren Sie SCA-Lösungen, die Teil eines ganzheitlichen AST-Portfolios sind.

Oder solche, die sich besonders gut mit Ihren vorhandenen Systemen integrieren lassen.

Prüfen Sie Integrationsoptionen mit Ihren Package-Managern, Build-Tools, Code-Repositories und Bug-Tracking-Lösungen.

Favorisieren Sie Lösungen mit produkt-übergreifenden Synergien, um die Behebung zu beschleunigen und zuverlässige und handlungsrelevante Ergebnisse zu erhalten.

Achten Sie darauf, dass sich die Lösung mit vorhandenen SDLC- und CI-/CD-Tools integrieren lässt.

Sie müssen in der Lage sein, automatisiert Scans anzustoßen und Ergebnisse zu teilen, um die Fehlerbehebung zu beschleunigen.

Prüfen Sie, ob die Lösung zentralisierte Tools für das User-Management, das Anlegen von Projekten und das Anstoßen von Scans in unterschiedlichen Analyselösungen unterstützt.

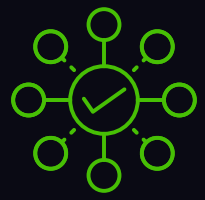
Lösungen, die dazu in der Lage sind, haben sich als besonders effizient bewährt und garantieren die attraktivste Cost-of-Ownership.

+ Fazit

Open-Source-Komponenten werden uns auch in Zukunft begleiten. Unternehmen sind daher gut beraten, SCA fest in ihrer Software-Security zu verankern.

Der Schlüssel zur erfolgreichen SCA-Implementierung ist es, eine Lösung zu wählen, die sich nahtlos in Ihre Entwicklungsumgebung einfügt, die interne und externe Risiko- und Compliance-Standards abbildet – und die den richtigen Mitarbeitern die richtigen Informationen zur richtigen Zeit an die Hand gibt.

Viele Security-Experten gehen davon aus, dass Angreifer in Zukunft verstärkt unsichere Open-Source-Libraries nutzen werden, um Zugriff auf wertvolle und sensible Daten zu erhalten. Hinzu kommt, dass die Nutzung von Open-Source-Komponenten rasant zunimmt, auch wenn deren Risikopotenzial in aller Regel nicht hinreichend dokumentiert, evaluiert und überwacht wird. Der Software Composition Analyse kommt daher schon heute eine Schlüsselrolle zu, und das wird sich auch in Zukunft nicht ändern.



+ Weiterführende Informationen:

Datenblätter

Zum Checkmarx CxSCA Datenblatt

E-Book

Gartner Magic Quadrant 2020 for Application Security Testing

Website

Open-Source-Security einer neuen Generation: Checkmarx CxSCA

Checkmarx

Über Checkmarx

Checkmarx entwickelt Software-Security für kritische Infrastrukturen, und setzt Maßstäbe, wenn es gilt, die Cyberrisiken von heute und morgen zu adressieren. Checkmarx bietet die branchenweit einzige vollumfängliche Software-Security-Plattform, die SAST, SCA, IAST und AppSec-Awareness-Trainings vereint, um Sicherheit in allen Phasen der CI/CD-Pipeline zu verankern und die Angriffsfläche zu minimieren. Über 1.400 Unternehmen weltweit setzen auf Checkmarx, um schneller sicherere Software bereitzustellen, darunter über 40 Prozent der Fortune 100 und zahlreiche große staatliche Einrichtungen.

Mehr unter Checkmarx.com