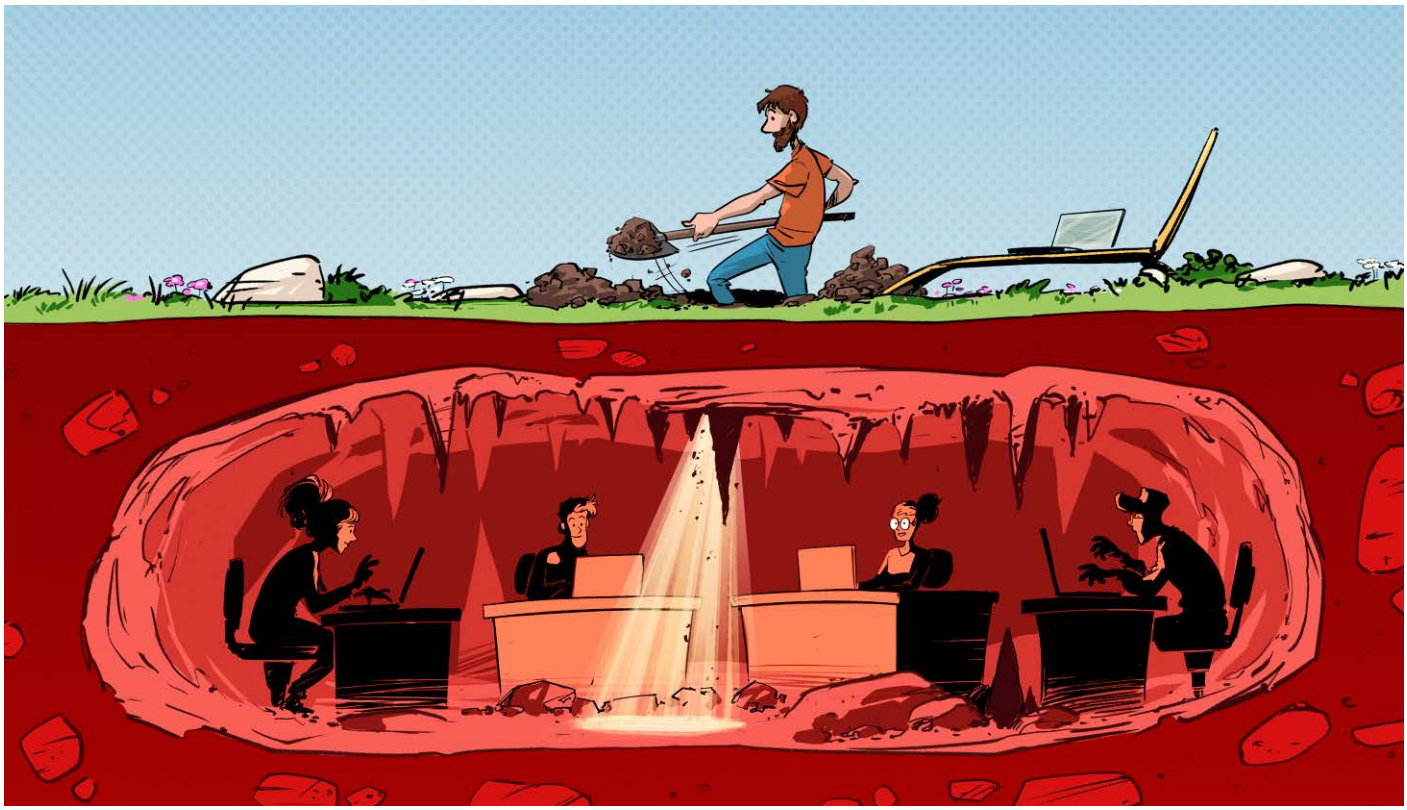


Bild: Albert Hulm

Herrscher über die Unterwelt

Abhängigkeiten systematisch erkennen und auswerten

Fast jede Software hängt von anderen ab. Wer da den Überblick verliert, handelt sich massenweise Probleme etwa in Gestalt von Sicherheitslücken ein. Software-Teilelisten, sogenannte SBOMs, versprechen Einblicke. Die volle Kontrolle bekommen Admins und Entwickler – na klar – mit einem weiteren Werkzeug: Dependency-Track.

Von Manuel Ottlik

Die Sicherheitslücke in der Java-Log-Bibliothek Log4j löste Ende 2021 Schockwellen in vielen Unternehmen aus. Eine harmlose Logging-Bibliothek konnten Angreifer leicht dazu überreden, Schadcode aus dem Internet zu laden. Was Admins und Beobachter überraschte, war nicht nur, dass eine ziemlich leicht zu missbrauchende Funktion so lange in einer so verbreiteten Bibliothek wie Log4j überleben konnte. Überraschend war vor allem das Treiben in vielen IT-Abteilungen – hektisch wurden vielerorts per Hand Tabellen mit verwendeter Software zusammengetragen und auf den Suchbegriff „Java“ gefiltert. Dann begann eine mühsame Suche, bei eigener Software im Quellcode, bei gekaufter Software in Dokumentationen und mit Mails an den Hersteller. Die verzweifelten Fragen: „Wo zur Hölle könnte in unserem Netzwerk Log4j stecken? Haben wir vielleicht etwas übersehen?“ Probleme, die sich nicht auf Mittelständler mit Admin-Einzelkämpfern beschränkten. Unsere Redaktion erreichten auch Berichte, wie die IT-Abteilungen großer Konzerne mit der Frage umgingen. Selbst wenn es einen CISO (Chief Information Security Officer) und ganze Teams für IT-Sicherheit gab, gingen interne Mails an alle Abteilungen und ihre Leitungen. Die sollten schnellstmöglich prüfen und bestätigen, dass die eingesetzten Anwendungen frei von Log4j oder abgesichert seien.

Was sich all diese Softwarebetreiber in diesem Moment wünschten: eine zentrale interne Datenbank, die sämtliche Abhängigkeiten aller Anwendungen kennt. Am besten natürlich durchsuchbar und mit einer Anbindung an Sicherheitslückendatenbanken. In einer perfekten Welt hätte man nur „log4j“ in ein Suchfeld

eintippen müssen, um sämtliche Vorkommen der betroffenen Bibliothek im eigenen Betrieb zu finden – feinsäuberlich aufgeschlüsselt nach Projekt und mit der jeweiligen Version. Für einzelne Programmiersprachen gibt es bereits Ansätze, die Abhängigkeiten eines Projekts auf Schwachstellen überprüfen. In der JavaScript-Welt greift man etwa zum Befehl `npm audit`. Doch damit ist einer IT-Abteilung, die außerdem noch Docker-Container, .NET, Java-Applikationen und Python-Skripte im Einsatz hat, nur bedingt geholfen. Kaum jemand wird ausschließlich Anwendungen in einer einzigen Programmiersprache nutzen und mit vielen fragmentierten Informationsquellen ist niemandem geholfen.

Software Bill of Materials

Man braucht etwas, das unabhängig von einer bestimmten Sprache oder einem Framework funktioniert. In anderen Industriezweigen gibt es das schon seit Langem: eine strukturierte Materialliste, eine „Bill of Materials“. So etwas gibt es auch für Software, die sogenannte „Software Bill of Materials“, kurz SBOM. Ähnlich wie bei der Zutatenliste auf dem Lieblingsmüsli enthält eine SBOM alle Komponenten, aus denen eine Software besteht – und zwar in einem standardisierten und maschinenlesbaren Datenformat. Darin stehen alle NPM-Pakete, die ein JavaScript-Frontend benötigt, und alle Pip-Pakete, die ein Python-Projekt voraussetzt.

International steigt die Verbreitung von SBOMs aktuell: In den USA sind sie durch eine Executive Order des Weißen Hauses im Mai 2021 verpflichtend für alle Softwarehersteller geworden, die an Behörden liefern (siehe ct.de/y9wt) – im deutschen IT-Grundschutz-Kompendium ergibt die Suche nach „SBOM“ und ähnlichem aktuell dagegen keine Treffer.

Die populärste Spezifikation für SBOMs heißt CycloneDX, ein Open-Source-Projekt, hinter dem die in Sicherheitsfragen aktive OWASP Foundation steckt. CycloneDX spezifiziert eine Datei, die alle Bestandteile einer Software strukturiert auflistet. Sie kann in XML oder JSON verfasst werden und setzt auf bereits bestehenden Standards wie „Package URL“ (PURL) und „Common Platform Enumeration“ (CPE) auf, um die Komponenten selbst eindeutig zu identifizieren. Informationen zu beiden finden Sie über ct.de/y9wt. Mit dem CycloneDX-SBOM-Standard können Anwendungen, Container, Betriebssysteme und Firmware sowie Frameworks und Bibliotheken beschrieben werden. Außerdem gibt es im CycloneDX-Kosmos noch andere BOMs aus dem IT-Kontext, beispielsweise für Software-as-a-Service (SaaSBOM).

Ein simples CycloneDX-Dokument im JSON-Format sieht zum Beispiel wie folgt aus:

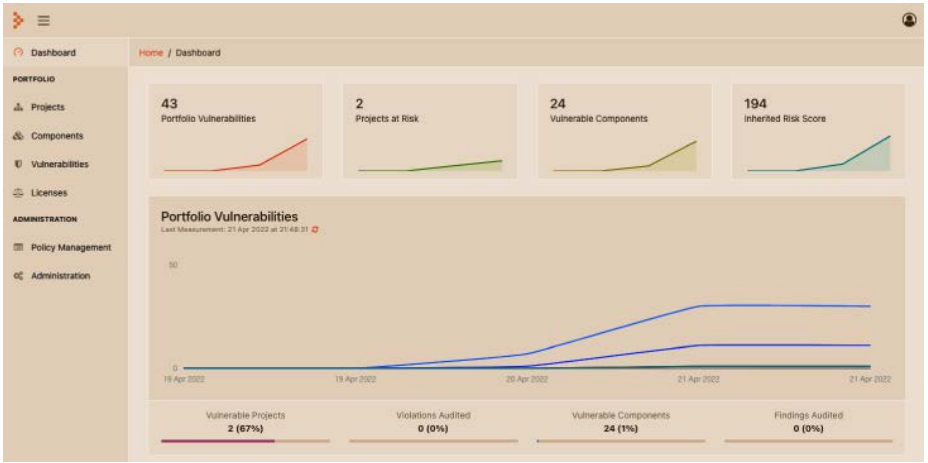
```
{
 "bomFormat": "CycloneDX",
 "specVersion": "1.2",
 "version": 1,
 "components": [
  {
   "name": "example-library",
   "group": "example-user",
   "type": "library",
   "version": "1.3.5",
   "description": "Just an example",
   "licenses": [
    { "license": {"id": "MIT"} }
   ],
   "purl":"pkg:composer\/example-user
           \/example-creator@1.3.5"
  }
 ]
}
```

Es beginnt mit Metadaten, danach folgt unter dem Schlüssel `components` eine Liste von Bestandteilen. Für jede Komponente wird verpflichtend ein Name und ein Typ angegeben. Weitere Angaben zur Identifikation sind laut Standard optional, aber ratsam, damit Pakete später eindeutig und automatisch identifiziert werden können. Die PURL im obigen Beispiel verrät, dass es sich um ein Paket aus dem PHP-Paketmanager Composer handelt.

Neben Paketen darf eine SBOM auch externe Dienste auflisten, auf die die Software zugreift. Wenn Ihre Software also zum Beispiel Wetterdaten von einem externen API bezieht, würde sich diese Schnittstelle unter `services` wiederfinden. Ähnlich einer Lieferkette in der Industrie kann eine SBOM auch eine Baumstruktur mit weiteren indirekten Abhängigkeiten enthalten. Wenn bei der Erstellung der SBOM bereits Schwachstellen bekannt sind, können diese unter `vulnerabilities` abgelegt werden. Welche Datenfelder CycloneDX darüber hinaus noch bereithält, verrät die gut aufbereitete CycloneDX-Dokumentation (zu finden über ct.de/y9wt).

Übersicht mit Dependency Track

Falls Sie jetzt als Entwickler die Angst überkommt, künftig neben Programmcode, Unit Tests und Dokumentation auch noch SBOMs schreiben zu müssen, seien Sie beruhigt: Diese Fleißarbeit übernehmen Kommandozeilenwerkzeuge, die es für alle gängigen Programmiersprachen gibt. Dazu später mehr – zunächst zur Perspektive der Admins, die von ihren Softwareherstellern SBOMs bekommen haben. Dass Hersteller SBOMs bereitstellen, ist heute noch keinesfalls selbstverständlich und bisher haben sich auch keine zentralen Anlaufstellen wie ein „SBOM-Hub“ zum Download der Dokumente etabliert. Es bleibt nur ein Blick in die Dokumentation oder bei Kaufsoftware eine vertragliche Vereinbarung mit dem Hersteller. Je mehr Kunden es schaffen, ihre Hersteller zu überzeugen, desto schneller kann sich der Standard durchsetzen.



Mit einem Blick wissen Sie dank Dependency-Track, wie es um die Sicherheit Ihrer Systeme steht.

Keine Angst vor Log4j: Dependency-Track warnt aufgrund einer passenden Richtlinie vor problematischen Bibliotheksversionen in einer Anwendung.

Mit den Dokumenten in JSON oder XML allein ist Ihnen aber auch noch nicht viel geholfen; Sie brauchen ein Werkzeug, das die SBOMs einliest, die Komponenten auf Schwachstellen analysiert und die Ergebnisse anschaulich darstellt. Genau das macht Dependency-Track: Für jede Software, die in einer Organisation verwendet wird, legt man künftig in Dependency-Track ein Projekt an, hinterlegt Software, Versionsnummer und SBOM und bekommt Übersichten über den aktuellen Zustand schön aufbereitet in einem Web-Frontend.

Dort finden Sie Tabellen zu verwendeten Bausteinen, deren Lizenzen sowie Baumstrukturen mit weiteren Abhängigkeiten. Die wichtigste Funktion ist aber die Analyse der Komponenten auf bekannte Schwachstellen. Abgeglichen wird einmal täglich gleich mit mehreren Schwachstellendatenbanken: der National Vulnerability Database (NVD), dem Sonatype OSS Index, den NPM Audit Advisories, der GitHub Advisory Database und optional auch mit dem kostenpflichtigen Dienst VulnDB. Alle Schwachstellen können Sie in einer Tabelle in Dependency-Track einsehen und erfahren auch direkt, welche Ihrer Projekte von der Schwachstelle betroffen sind. Damit Sie im Ernstfall schnell informiert werden, kann die Software E-Mails oder Benachrichtigungen für Microsoft Teams, Slack oder sonstige Webhook-fähige Systeme versenden.

Dependency Track zeigt außerdem an, ob es für eine verwendete Komponente schon eine neuere Version ohne Schwachstelle gibt, die Sie verwenden könnten. Dafür werden die Paketmanager der unterschiedlichen Programmiersprachen abgefragt. Das funktioniert sogar für private Pakete, die Sie in Ihrer firmeninternen Paket-Registry liegen haben. Diese können Sie als zusätzliche Quelle in Dependency-Track hinterlegen.

Die Plattform gleicht eingesetzte Software nicht nur gegen externe Datenbanken ab, sondern erlaubt auch eigene Richtlinien (Policies) zu erstellen, die aus mehreren Bedingungen bestehen dürfen. So können Sie etwa überprüfen lassen, ob eine Anwendung Schwachstellen eines gewissen Schweregrads hat oder eine bestimmte Komponente in einer problematischen Version enthält. Auf dem Screenshot oben sehen Sie, wie Sie mit einer einfachen Policy alle Anwendungen identifizieren, die Log4j mit der berühmten Sicherheitslücke nutzen. Wenn die Bedingungen zutreffen, wird eine „Policy Violation“ an der Anwendung markiert. Jeder Policy können Sie dabei auch einen Schweregrad zuweisen, der bei Verstoß aufleuchtet. Das funktioniert nicht nur für Sicherheitsprobleme, sondern auch für Lizenz-Informationen, die an jeder Komponente in der SBOM vermerkt sind. So können Sie zum Beispiel erkennen, wenn eine Bibliothek oder Software ihr Lizenzmodell ändert und plötzlich zum Problem wird.

Live und in Farbe

Viele dieser Funktionen richten sich zweifelsohne an große Organisationen, die Mitarbeiter nur für solche Sicherheits- und Lizenzfragen beschäftigen können. Auch an LDAP-Anbindung und Berechtigungsverwaltung ist gedacht. Dass Dependency-Track so viele Funktionen mitbringt, sollte Sie aber auch in einer kleineren Umgebung nicht abschrecken, die Software auszuprobieren und zu nutzen. Installation und Einstieg sind nämlich denkbar einfach. In wenigen Minuten fahren Sie die Umgebung als lokale Umgebung mit Docker hoch (eine ausführliche Installationsanleitung finden Sie über ct.de/y9wt).

Die Dependency-Track-Entwickler bieten eine Docker-Compose-Datei an, mit der Sie die Plattform ausprobieren können. Laut Doku werden mindestens 4,5 GByte Arbeitsspeicher und zwei CPU-Kerne gefordert, im Test kamen die beiden Container aber auch mit deutlich weniger RAM klar. Laden Sie sich die Docker-Compose-Datei herunter (siehe ct.de/y9wt) und starten Sie die Software mit:

```
docker compose up -d
```

Unter der Adresse http://localhost:8080 finden Sie dann das Frontend. Mit dem Benutzernamen `admin` und dem initialen Passwort `admin` können Sie sich anmelden. Nachdem Sie das Passwort geändert haben, wählen Sie im Menü „Projects“ aus und erstellen Sie mit dem Button „Create Project“ ein neues Projekt. Wenn Sie im Reiter „Components“ unter „Upload BOM“ eine SBOM-Datei im CycloneDX-Format hochladen, wird sie anschließend analysiert.

Falls Ihre Softwarelieferanten noch keine SBOMs bereitstellen und Sie zum Ausprobieren von Dependency-Track auch keine selbst generieren möchten, finden Sie eine Auswahl an Beispielen zum Download über ct.de/y9wt. Mit einem geladenen BOM und echten Sicherheitsproblemen bekommen Sie schnell ein Gefühl, was mit Dependency-Track möglich ist und welche Funktionen Sie im Alltag nutzen möchten.

SBOMs generieren

Der folgende Abschnitt richtet sich an alle, die selbst Software entwickeln, vertreiben oder vermieten und SBOMs im CycloneDX-Format anbieten möchten oder sollen. Die gute Nachricht: Rund um das noch recht junge Format – CycloneDX 1.0 erschien im März 2018 – hat sich schnell eine große Open-Source-Gemeinschaft gebildet, die unter anderem SBOM-Generatoren für unterschiedlichste Programmiersprachen und Techniken bereitstellt. Die funktionieren nicht für jede Sprache identisch: Für Node.js-Projekte werden SBOMs über einen Kommandozeilenaufruf generiert, für Java-Projekte gibt es ein Maven-Plug-in, das Sie Ihrem Projekt hinzufügen müssen. Nicht alle Generatoren setzen den CycloneDX-Standard schon komplett um: Das CLI für Node.js unterstützt ak-

tuell noch keine Dependency-Trees und das Werkzeug für PHP-Projekte hat gerade einmal 15 Sterne auf GitHub, steht also noch ganz am Anfang seiner Karriere. Beide erzeugen aber gültige SBOMs, die Dependency-Track auswerten kann. Wenn Sie auf der Suche nach einem Generator für Ihre Programmiersprache sind, ist das „Tool Center" von CycloneDX, eine Übersicht über alle Projekte, die erste Anlaufstelle, zu finden unter cyclonedx.org/tool-center.

Generieren ganz praktisch

Um ein Gefühl zu bekommen, wie wenig Arbeit das Generieren von SBOMs macht, können Sie die Arbeitsschritte an einem Beispiel nachvollziehen. Dafür haben wir ein Projekt aus einem JavaScript-Artikel von Ende 2020 herausgesucht, der schon länger nicht aktualisiert wurde – die Chancen, Sicherheitslücken zu finden, stehen also gut. Den Code bekommen Sie über ct.de/y9wt aus einem GitHub-Repository, die fertige Datei liegt da bereits unter 02-application/sbom.json. Über das Tool Center finden Sie schnell ein geeignetes Werkzeug für JavaScript, installiert wird es mit:

```
npm install -g @cyclonedx/bom
```

Laden Sie anschließend das Beispiel-Repository oder ein anderes JavaScript-Projekt herunter und navigieren auf der Kommandozeile in den Ordner, in dem die Datei package.json liegt. Den Generator starten Sie dann mit

```
cyclonedx-node --include-dev \
--include-license-text \
--type application \
--output bom.json
```

Die erzeugte Datei können Sie anschließend in Dependency-Track hochladen. Die Plattform verarbeitet die SBOMs aber asynchron – wundern Sie sich also nicht, wenn nicht augenblicklich die Alarmglocken schrillen. Nach ein paar Minuten zeigt Dependency-Track die Anzahl der Schwachstellen an, aufgeteilt nach Schweregrad. Im Tab „Audit Vulnerabilities" sehen Sie eine Liste aller Schwachstellen, die Dependency-Track beim Abgleich mit den Datenbanken identifiziert hat.

Dependency-Track findet für das Beispielprojekt eine kritische Sicherheitslücke in der Bibliothek lodash. Wenn Sie in Zukunft immer direkt informiert werden wollen, wenn diese problematische Bibliothek irgendwo auftaucht, legen Sie sich eine Policy dafür an: Wechseln Sie dafür im Menü auf „Policy Management" und klicken dort auf „Create Policy". Vergeben Sie einen Namen und erstellen Sie anschließend eine Bedingung für die sogenannten „Coordinates" (den vollen Namen einer Bibliothek), in diesem Fall „lodash > 4.0.0". Weil auch die Policy Violations asynchron geprüft werden, müssen Sie ein paar Minuten warten – dann sehen Sie an dem erstellten Projekt unter dem Tab „Policy Violations" sieben Verstöße gegen Ihre neue Richtlinie. Auf dem gleichen Wege können Sie Pakete auf deren Lizenz überprüfen: Erstellen Sie zum Beispiel eine Policy mit dem Namen „keine OS-Lizenz", verändern Sie den Operator der Bedingungen auf „All" und nehmen Sie in den Bedingungen mit „Licence / is not" alle Lizenz-Arten auf, die Sie akzeptieren möchten. Für jede Komponente, die keine dieser Lizenzen verwendet, wird dann eine Violation generiert. Wenn Sie nicht auf das nächste Prüfintervall warten wollen, können Sie die SBOM einfach erneut hochladen – dann startet Dependency-Track die Analyse auch außerhalb des Intervalls.

Continuous Security Monitoring

Als Entwickler könnten Sie Ihre generierte SBOM jetzt irgendwo hochladen, wo Ihre Nutzer sie finden und bestenfalls auch per Skript herunterladen können; in der Dokumentation können Sie auf den Link verweisen. Vom CycloneDX-Projekt gibt es auch einen einfachen Repository-Server zum Selbsthosten, auf den Sie SBOMs hochladen können (zu finden im Tool Center). Es reicht aber auch ein beliebiger Webserver. Sobald Sie den Prozess der Bereitstellung einmal per Hand durchgespielt haben, sollten Sie die Schritte umgehend automatisieren. Das Generieren von SBOMs können Sie am besten einer CI/CD-Umgebung wie GitLab oder GitHub Actions überlassen. Dann können Sie sicher sein, dass die SBOM mit jeder Änderung am Code aktualisiert wird. Für CI/CD-Integrationen finden Sie auf der CycloneDX-Seite ebenfalls Projekte. Die sind aber teilweise noch etwas spärlich und unflexibel. Einfacher ist es meist, den Befehl für einen passenden Generator selbst in eine CI/CD-Aktion zu verpacken und auch den Upload auf Ihren Server einzubauen.

Wenn Sie als Softwarenutzer einen Weg finden, SBOMs automatisiert zu beziehen (bei eigener Software aus Ihrer CI/CD-Umgebung oder mit geskripteten Downloads von Herstellern) und diese vollautomatisch zu Dependency-Track hochladen, können Sie stolz erzählen, dass Ihre Organisation „Continuous Security Monitoring" (CSM) umsetzt. Damit SBOM-Dateien automatisch bei Dependency-Track ankommen, nutzen Sie dessen HTTP-API. Die Plattform bietet dafür für jede Gruppe von Benutzern API-Schlüssel an, die man bei API-Aufrufen übermitteln muss. Diese Schlüssel erzeugen Sie in der Weboberfläche über Administration/Access Management/Teams. Der Upload frischer SBOMs ist dann nur noch ein einfacher HTTP-PUT-Befehl, die Dokumentation erklärt diesen Schritt ausführlich (siehe ct.de/y9wt) und zeigt auch, wie man ihn aus einer CI/CD-Umgebung absetzt.

Fazit

Eine Software Bill of Material (SBOM) und CycloneDX trennen die Beschreibung der Anwendung und ihrer Abhängigkeiten von einer konkreten Programmiersprache. Der Standard unterstützt alle gängigen Programmiersprachen, Docker-Container und weitere Artefakte wie Helm-Charts und sogar externe Dienste. Also vieles von dem, was Sicherheitslücken haben kann. Ohne eine Software wie Dependency-Track hat man an den JSON- oder XML-Dateien aber keine Freude. Sie bereitet die SBOMs visuell auf, prüft täglich die Sicherheitslage und pustet im Falle einer Sicherheitslücke Benachrichtigungen über diverse Kanäle raus.

Um SBOMs erfolgreich in einer Organisation einzuführen, sind aber auch nicht-technische Probleme zu lösen: Richtig sinnvoll ist das Unterfangen erst, wenn wirklich alle Anwendungen ohne Ausnahme erfasst werden. Davon muss man sowohl alle Entwickler eigener Anwendungen als auch Lieferanten weiterer Software überzeugen. Das ist aktuell harte Arbeit und ein langfristiges Projekt, das sich aber lohnen kann: Bei der nächsten großen Sicherheitslücke kennen Sie die Auswirkungen auf Ihre Organisation ohne panische Rundmails und hastig erzeugter Excel-Tabellen. Genug Arbeit wartet aber auch dann noch. Nachdem Dependency-Track Alarm geschlagen hat, muss man die Probleme schließlich noch abstellen.

(jam@ct.de) c't

Dokumentationen und Werkzeuge: ct.de/y9wt