Supply-Chain-Attacken

So wird Ihre Software-Lieferkette gehackt

Software-Supply-Chain-Attacken stehen bei Cyberkriminellen hoch im Kurs. Das sollten Sie zum Thema wissen.

(Software-)Supply-Chain-Attacken verteilen Schadcode über Kunden- und Partnernetzwerke und können massiven Schaden verursachen. (Foto: fotogestoeber – shutterstock.com)

Angriffe auf Software-Lieferketten haben dieses Jahr immer wieder für Schlagzeilen gesorgt. Doch Supply-Chain-Attacke ist nicht gleich Supply-Chain-Attacke.

Software-Supply-Chain-Attacke – Definition

Der Begriff Supply-Chain-Attacke meint Angriffsszenarien, bei denen Cyberkriminelle in den Herstellungsprozess, beziehungsweise den Entwicklungslebenszyklus eingreifen oder ihn kapern, so dass mehrere Endverbraucher des fertigen Produkts oder Dienstes nachteilig beeinflusst werden. Im Zusammenhang mit Softwareentwicklung kann das passieren, wenn:

- Code-Bibliotheken oder einzelne Komponenten innerhalb eines Software-Builds verfälscht,
- Software-Update-Binärdateien trojanisiert,
- Code-Signatur-Zertifikate gestohlen oder
- SaaS-Hostingserver kompromittiert werden.

Bei jedem Angriff auf die Software-Lieferkette schalten sich die Angreifer dann entweder im Vorfeld oder in der Mitte der Lieferkette ein, um ihre maliziösen Aktivitäten und deren Folgen auf möglichst viele Benutzer loszulassen. Im Vergleich zu isolierten Sicherheitsverletzungen weisen erfolgreiche Supply-Chain-Angriffe im Regelfall ein wesentlich größeres Ausmaß als auch weitreichendere Auswirkungen auf.

Supply-Chain-Angriffe – Beispiele

Die folgenden sechs Beispiele zeigen, welche Methoden Cyberkriminelle bei Supply-Chain-Attacken in der Praxis eingesetzt haben.

1. Upstream Server Compromise

Bei den meisten Angriffen auf die Software-Lieferkette dringt ein Angreifer in einen Upstream-Server oder ein Code-Repository ein und injiziert bösartigen Code. Diese Nutzlast wird dann nachgelagert an viele Benutzer verteilt. Aus technischer Sicht ist das jedoch nicht immer der Fall.

Die Codecov-Supply-Chain-Attacke ist ein solches Beispiel. Obwohl der Vorfall oft mit dem SolarWinds-Angriff verglichen wird, gibt es gewichtige Unterschiede zwischen den beiden Vorfällen. Der Angriff auf die Lieferkette von SolarWinds war das Werk raffinierter Bedrohungsakteure, die eine legitime Update-Binärdatei veränderten. Wie bereits von FireEye analysiert, lauert der bösartige Code dabei in einer gefälschten DLL. Diese Methode ruft die HTTP-basierte Backdoor auf, wenn die Orion Software das Inventory Manager Plugin lädt. Der SolarWinds-Upstream-Angriff kam jedoch erst voll zum Tragen, als die veränderte Binärdatei ihren Weg zu über 18.000 SolarWinds-Kunden, darunter auch US-Regierungsbehörden, fand.

Im Fall des schlagzeilenträchtigen Lieferkettenangriffs auf den US-IT-Dienstleister Kaseya gelang es den Angreifern, verschiedene Zero Day Exploits in der IT-Management-Software VSA auszunutzen und so bösartige Updates – und damit die REvil Ransomware – an die Kunden des Unternehmens zu verteilen. Bei diesen Kunden handelte sich vor allem um Managed Service Provider, die wiederum die Netzwerke hunderter Unternehmen betreuen:

Im Fall von Codecov wurde jedoch kein Schadcode per Downstream verteilt. Die Angreifer hatten sich über einen fehlerhaften Docker-Image-Erstellungsprozess Zugangsdaten verschafft. Mit diesen gelan es ihnen dann, den Bash Uploader zu modifizieren, um Umgebungsvariablen zu sammeln, die von den CI/CD-Umgebungen der Kunden übertragen werden, wie es im offiziellen Security Advisory heißt. Tatsächlich sollen die Codecov-Angreifer Hunderte von Kundennetzwerken mit den Zugangsdaten, die sie über den gehackten Bash Uploader gesammelt hatten, angegriffen haben. Kurze Zeit später gab HashiCorp bekannt, dass der Codecov-Vorfall zur Offenlegung seines privaten PGP-Schlüssels geführt hat. Dieser wird verwendet, um Softwarepakete zu signieren und verifizieren.

2. Midstream Server Compromise

Der Begriff „Midstream" bezieht sich auf Fälle, in denen Angreifer eine zwischengeschaltete Software-Upgrade-Funktionalität oder ein CI/CD-Tool kompromittieren und nicht die ursprüngliche Upstream-Quellcode-Basis. Letzten Monat informierte Click Studios, Hersteller des bei vielen Fortune-500-Unternehmen beliebten Passwort-Managers Passwordstate, seine Kunden über einen Supply-Chain-Angriff. Dabei missbrauchten die Angreifer die „In-Place-Upgrade"-Funktionalität, um bösartige Updates an Passwordstate-Benutzer zu verteilen. Die illegalen Updates enthielt eine modifizierte DLL-Datei.

In einem Sicherheitshinweis erklärte Click Studios: „Die Kompromittierung bestand etwa 28 Stunden lang. Wir nehmen an, dass nur Kunden betroffen sind, die In-Place-Upgrades zwischen den oben genannten Zeitpunkten durchgeführt haben. Manuelle Upgrades von Passwordstate sind nicht kompromittiert. Die Passwörter betroffener Kunden können abgegriffen worden sein."

Wenig überraschend folgten Phishing-Attacken gegen Click-Studios-Benutzer, in denen Angreifer Links zu einer aktualisierten Malware-Version einfügten. Dieser Angriff auf die Lieferkette hatte nicht nur einen technischen (der Upgrade-Prozess wurde manipuliert), sondern auch einen Social-Engineering-Aspekt. Den Angreifern war es gelungen, die Benutzerhandbücher, Hilfedateien und PowerShell-Erstellungsskripte in der gefälschten Update-Zip-Datei so zu ändern, dass sie auf ihren bösartigen CDN-Server verweisen.

Das zeigt eine weitere Schwachstelle auf: Insbesondere neue Entwickler oder Software-Konsumenten stufen Links zu Content Distribution Networks (CDNs) nicht immer als verdächtig ein. Schließlich nutzen Softwareanwendungen und Websites CDNs, um Updates, Skripte und andere Inhalte bereitzustellen. Kreditkarten-Skimming-Angriffe wie Magecart sind ein weiteres Beispiel für diese Art von Lieferkettenangriff. Bei einigen Angriffen wurden Amazon CloudFront CDN-Buckets kompromittiert, um den bösartigen JavaScript-Code an eine größere Anzahl von Web-

sites zu verteilen, die auf solche CDNs angewiesen sind.

3. Dependency-Confusion-Angriffe

Ihre simple und automatisierte Natur machen Dependency-Confusion-Angriffe so gefährlich. Sie funktionieren nämlich mit minimalem Aufwand auf Seiten des Angreifers und nutzen eine inhärente Designschwäche mehrerer Open-Source-Ökosysteme aus.

› (Foto: RadiasaTutorial Rti – Vecteezy.com)

Vereinfacht ausgedrückt, funktioniert Dependency Confusion (oder Namespace Confusion), wenn ein Software-Build eine private, intern erstellte Abhängigkeit verwendet, die nicht in einem öffentlichen Repository existiert. Ein Angreifer ist in der Lage, eine gleichnamige Abhängigkeit in einem öffentlichen Repository zu registrieren, mit einer höheren Versionsnummer. Dann wird sehr wahrscheinlich die (öffentliche) Abhängigkeit des Angreifers mit der höheren Versionsnummer in Ihren Software-Build gezogen.

Der Ethical Hacker Alex Birsan nutzte die Lücke in populären Ökosystemen wie PyPI, npm und RubyGems aus und war so in der Lage, sich in 35 große Tech-Unternehmen zu hacken. Wenige Tage nach der Veröffentlichung seiner Forschungsergebnisse wurden diverse Ökosysteme mit Nachahmerpaketen überflutet.

Es gibt mehrere Möglichkeiten, das Verwirrspiel um Abhängigkeiten aufzulösen. Dazu gehört vor allem, die Namen all Ihrer privaten Abhängigkeiten in öffentlichen Repositories zu registrieren, bevor ein Angreifer das tut. Darüber hinaus empfiehlt es sich, automatisierte Lösungen – beispielsweise eine Software Development Lifecycle Firewall – einzusetzen, die verhindern, dass widersprüchliche Abhängigkeiten Ihre Lieferkette beeinträchtigen.

4. Gestohlene Zertifikate

SSL/TLS-Zertifikate sind im Netz allgegenwärtig und schützen die Online-Kommunikation. Wird der private Schlüssel eines SSL-Zertifikats kompromittiert, wird daraus nichts.

Im Januar 2021 gab Mimecast bekannt, dass ein Zertifikat kompromittiert wurde, das von seinen Kunden verwendet wird, um eine Verbindung zu Microsoft-365-Exchange-Diensten herzustellen. Das wirkte sich auf die Kommunikation von etwa 10 Prozent der Mimecast-Nutzer aus. Obwohl das Unternehmen nicht ex-

plizit bestätigte, dass es sich um ein SSL-Zertifikat handelte, liegt diese Vermutung Security-Forschern zufolge nahe.

Während ein kompromittiertes SSL-Zertifikat problematisch ist, kann ein gestohlenes Code-Signing-Zertifikat (also ein kompromittierter privater Schlüssel) weitreichendere Folgen für die Softwaresicherheit haben: Angreifer, die diesen Schlüssel in die Hände bekommen, können ihre Malware möglicherweise als legitime Software oder offizielles Update signieren. Aus diesem Grund ist auch das bereits erwähnte Beispiel der Preisgabe des privaten PGP-Schlüssels von HashiCorp im Rahmen des Codecov-Lieferkettenangriffs problematisch. Obwohl es bisher keine Hinweise darauf gibt, dass der kompromittierte Schlüssel von Angreifern zum Signieren von Malware missbraucht wurde.

5. Angriffe auf CI/CD Pipelines

Sonatype beobachtete kürzlich einen vielschichtigen Angriff auf die Software-Lieferkette, Dieser beinhaltete nicht nur bösartige Pull-Requests im GitHub-Projekt eines Benutzers, sondern missbrauchte auch die CI/CD-Automatisierungsinfrastruktur von GitHub, GitHub Actions, um Kryptowährung zu schürfen.

Die Angreifer klonten legitime GitHub-Repositories, die das GitHub-Action-Skript im Repository leicht veränderten und reichten einen Pull-Request für den Projektinhaber ein, um diese Änderungen wieder in das ursprüngliche Repository einzubinden. Sollte ein Projektinhaber den geänderten Pull-Request beiläufig genehmigen, ist der Supply-Chain-Angriff gelungen. Ein solcher Angriff stützt sich auf zwei Komponenten: Entweder man bringt einen Entwickler dazu, einen bösartigen Pull-Request zu akzeptieren. Oder die vorhandene, automatisierte CI/CD-Infrastruktur wird für maliziöse Aktivitäten zweckentfremdet.

❯ (Foto: RadiasaTutorial Rti – Vecteezy.com)

Sicherheitsforscher konnten erfolgreich in Domänen der Vereinten Nationen (UN) eindringen und auf über 100.000 UNEP-Mitarbeiterdatensätze zugreifen. Das war möglich, weil sie auf diesen Domänen exponierte Git-Ordner und „git-credentials"-Dateien gefunden hatten. Fallen Git-Anmeldeinformationen

einem Bedrohungsakteur in die Hände, kann dieser nicht nur private Git-Repositories klonen, sondern möglicherweise auch bösartigen Code einschleusen, um eine Supply-Chain-Attacke zu initiieren.

Bisher lag der Fokus vor allem darauf, Entwicklern sichere Coding-Praktiken oder DevSecOps-Automatisierungstools zu empfehlen. Ebenso wichtig ist jedoch inzwischen auch die Absicherung von CI/CD-Pipelines, Cloud-nativen Containern und ergänzenden Entwickler-Tools und -Infrastrukturen.

6. Social Engineering

Die Linux Foundation hat kürzlich Forscher der University of Minnesota „verbannt", weil diese absichtlich fehlerhafte „Patches" vorgeschlagen hatten, die Schwachstellen in den Linux-Kernel-Quellcode einführten.

Obwohl dieser Fall aufgeflogen ist, zeigt er die Realität: Entwickler sind rar gesät und haben nicht immer die Zeit, jeden einzelnen Code-Commit oder Änderungen zu überprüfen. Dazu kommt, dass Social Engineering auch unverdächtigen Quellen entspringen kann – in diesem Fall von vermeintlich glaubwürdigen Wissenschaftlern mit „.edu"-E-Mail-Adresse.

Supply-Chain-Attacken – das Risiko steigt

All diese Beispiele aus der Praxis zeigen verschiedene Schwachstellen, Angriffsvektoren und Techniken, die Bedrohungsakteure bei erfolgreichen Angriffen auf die Lieferkette einsetzen. Da sich diese Angriffe immer weiterentwickeln und Herausforderungen darstellen, sind innovativere Lösungen und Strategien erforderlich, wenn es um Software-Sicherheit geht.

Ax Sharma

Ax Sharma ist ein Security- und Technologieexperte und schreibt für die US-Schwesterpublikation CSO Online.