Netzbasierte Erkennung von mittels Port Knocking versteckten Diensten und Backdoors

8. DFN-Forum Kommunikationstechnologien
TK II: ITC Management & Sicherheit

Felix von Eye
Leibniz-Rechenzentrum
Boltzmannstr. 1
85748 Garching
Germany
+49 (0)89 35831-8876
Felix.vonEye@lrz.de

Michael Grabatin
Leibniz-Rechenzentrum
Boltzmannstr. 1
85748 Garching
Germany
+49 (0)89 35831-7832
Michael.Grabatin@lrz.de

PD Dr. Wolfgang Hommel
Leibniz-Rechenzentrum
Boltzmannstr. 1
85748 Garching
Germany
+49 (0)89 35831-7821
Wolfgang.Hommel@lrz.de

Netzbasierte Erkennung von mittels Port Knocking versteckten Diensten und Backdoors

Felix von Eye, Michael Grabatin, Wolfgang Hommel

Munich Network Management Team
Leibniz-Rechenzentrum (LRZ)
Boltzmannstr. 1
85748 Garching b. München
{voneye,grabatin,hommel}@lrz.de

Abstract: Port Knocking ist ein etabliertes Verfahren, um Serverdienste versteckt zu betreiben und nur bei Bedarf für autorisierte Benutzer zugänglich zu machen. Legitim wird es beispielsweise häufig eingesetzt, damit Systemadministratoren weltweit per SSH auf ihre Server zugreifen können, ohne dass der SSH-Dienst von der Allgemeinheit kontaktiert werden kann, so dass die Angriffsfläche klein bleibt. Allerdings können auch Angreifer, die ein System erfolgreich kompromittiert und eine Backdoor eingerichtet haben, Port Knocking einsetzen, um deren Existenz möglichst lange zu verbergen. In diesem Beitrag wird zum einen ein Verfahren vorgestellt, wie die Ergebnisse von Portscans mit den durch Netflow gewonnenen Flow Records korreliert und analysiert werden können, um die Grundlage für eine netzbasierte Erkennung von derart versteckten Backdoors zu legen. Zum anderen wird ein Ausblick darauf gegeben, welche für diesen Zweck interessanten neuen Möglichkeiten sich durch programmierbare Controller für Software-defined Networks ergeben. Das vorgestellte Verfahren wurde prototypisch implementiert und zunächst in einer Laborumgebung getestet; das bereits identifizierte Verbesserungspotential wird diskutiert und ein Ausblick auf den praktischen Einsatz gegeben.

1 Einleitung

Einige Dienste, z. B. Web- oder Mail-Server, müssen meist dauerhaft und für jeden im Internet erreichbar sein, um ihre Aufgabe zu erfüllen. Diese Dienste sind durch ihre weltweite Erreichbarkeit auch durch Portscanner wie nmap [Lyo08], ZMap [DWH13] oder Masscan [Gra14] auffindbar. Demgegenüber gibt es eine Reihe weiterer Dienste, zum Beispiel für Remoteverbindungen mittels SSH, die meist einen stark eingeschränkten Benutzerkreis haben. Diese Dienste müssen somit nicht direkt erreichbar sein, sondern es ist ausreichend, die notwendigen Ports nur dann zu öffnen, wenn der Dienst von autorisierten Nutzern auch wirklich benötigt wird.

Dieser Ansatz birgt eine Reihe von Vorteilen. Zum einen wird ein solcher Dienst von einem nach Schwachstellen suchenden Angreifer nicht gefunden, da durch Zero-Day-Exploits selbst voll gepatchte Systeme Ziel eines Angriffs werden können. Weiterhin wird jemanden, der das System scannt, nicht direkt gezeigt, welchem Zweck das System dient, so

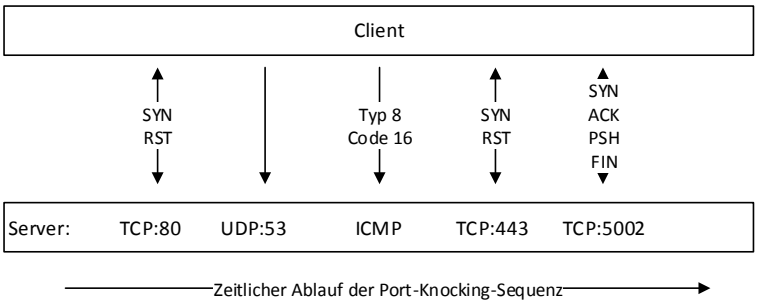

Abbildung 1: Beispiel einer Port-Knocking-Sequenz

dass auch unter Umständen Dienste betrieben werden können, die nicht offen propagiert werden sollen.

Eine häufig verwendete Variante, diese Funktion zu ermöglichen, ist Port Knocking. Das zu schützende System wird durch eine zusätzliche Software erweitert, die auch Verbindungsversuche zu geschlossenen Ports aufzeichnet und analysiert. Dies hat den Grund, dass für das Port Knocking beliebige Sequenzen von Verbindungsversuchen vordefiniert werden, die dann von der Port-Knocking-Software erkannt werden. Je nach Einstellung wird dann eine bestimmte Aktion auf dem System ausgeführt, wie beispielsweise an der Firewall einen speziellen Port für eine bestimmte anklopfende IP-Adresse zu öffnen oder ein Skript auszuführen, das den versteckten Dienst startet. Abbildung 1 zeigt als Beispiel eine Port-Knocking-Sequenz, bei der durch den Client nacheinander verschiedene spezielle Pakete verschickt werden: Port 80 mit TCP, Port 53 mit UDP, ein ICMP-Paket mit speziellen Header-Informationen, Port 443 mit TCP und schließlich ein vollständiger Verbindungsaufbau auf Port 5002 mit TCP.

Durch die große Zahl an möglichen Portnummern und die Verwendung verschiedener Protokolle (TCP, UDP oder ICMP) bzw. Flags (SYN, ACK, FIN, ...) in den Paketen der Aktivierungssequenz kann ein zufälliges Auslösen der Kombination praktisch ausgeschlossen werden. Weiterhin sind auch Wiederholungen einzelner Sequenzbestandteile möglich, was die Zahl der möglichen Kombinationen zusätzlich erhöht. Schließlich ist es auch denkbar, dass nicht immer die gleiche Sequenz zur Aktivierung verwendet wird, sondern eine Art One-Time-Sequenz in Analogie zu Einmalpassworten festgelegt wird, so dass selbst ein Angreifer, der den gesamten Verkehr zu dem System mithören kann, keine weiteren eigenen Verbindungen öffnen kann.

Ein Beispiel für eine Software, die Port Knocking auf Linux-Systemen implementiert, ist knockd [Vin14]. Darüber hinaus gibt es auch noch komplexere Methoden, die es einem Beobachter erschweren, Port Knocking zu erkennen, und es dem Benutzer erlauben, sich zu authentifizieren [VHLT07]. Es ist anzumerken, dass sich Port Knocking im Grenzbereich zwischen reiner Security-by-Obscurity und einem schwachen zusätzlichen Authentifizierungsmerkmal bewegt; dennoch erfreut es sich bei vielen Administratoren großer Beliebtheit, um die Hürden für Angreifer noch etwas höher zu heben.

Neben diesen zum Schutz von Systemen legitimierten Absichten, Dienste zu verstecken, gibt diese Technik aber auch Angreifern die Möglichkeit, längerfristig die Kontrolle über ein kompromittiertes System zu behalten. Ist es dem Angreifer erst mal gelungen, ein System erfolgreich vollständig zu kompromittieren, so ist er in der Lage, sich mit Hilfe von Rootkits weitestgehend einer Host-seitigen Erkennung zu entziehen. Über einen mit Port Knocking geschützten Remotedienst ist er darüberhinaus auch in der Lage, seine Kommunikation zu verschleiern. Dass dies keinesfalls nur eine theoretisch-akademische Bedrohung ist, zeigen folgende Beispiele.

Eines dieser Beispiele ist cd00r [oP00], eine in ca. 500 Zeilen C implementierte und ausführlich dokumentierte Backdoor, die auf eine vorher festgelegte Port-Knocking-Sequenz wartet und dann auf einem bestimmten Port einen Shell-Zugang öffnet. Eine Weiterentwicklung von cd00r ist SADoor [Nak03], welches mehr Konfigurationsmöglichkeiten durch eigene Konfigurationsdateien bietet. Einen etwas anderen Ansatz verfolgt das in C implementierte Programm evilshell [Sim08]; statt einen Port für eingehende Verbindungen zu öffnen, bietet die Backdoor eine Reverse-Shell, die sich nach einem speziellen eingehenden Paket zu dem Absender verbindet. Diese Beispiele für Malware sind unter Umständen nicht leicht zu entdecken, wie ein Bericht über die Analyse einer Backdoor, die bei einem Angriff auf das freenode-IRC-Netzwerk eingesetzt wurde, zeigt [Gro14].

Im Folgenden soll nun vorgestellt werden, wie man zum einen erkennen kann, dass versteckte Dienste bzw. mögliche Backdoors im eigenen Netz vorhanden sind. Da üblicherweise in einem Hochschulumfeld das Rechenzentrum zwar das Netz, aber nicht sämtliche Endgeräte administriert, liegt der Fokus dieser Arbeit auf der rein netzbasierten Erkennung. Im Forschungsprojekt SASER (Safe and Secure European Routing) ist das Ziel, Netz-Architekturen und Technologien für sichere zukünftige Netze zu entwickeln, wobei insbesondere Sicherheitsprobleme auf der IP-Schicht (Layer 3) adressiert werden sollen. Da insbesondere Router und Layer-3-fähige Switches zentrale Komponenten sind, die durch eine mögliche Kompromittierung das Ziel von sicheren Netzen nachhaltig stören können, ist es nötig, eine Backdoor-Erkennung in diesen Netzkomponenten zu entwickeln. Dieses Paper fokussiert dabei die oben erwähnten versteckten Dienste, die durch Port Knocking abgesichert sind. Schwachstellen oder eingebaute Hintertüren in regulären Diensten wie der Autorisierungskomponente bleiben daher unbetrachtet, da dies fundamental andere Erkennungsalgorithmen benötigt. Eine wirksame und im Bereich von Hochsicherheitsnetzen sinnvolle Gegenmaßnahme ist das Betreiben einer auf Whitelisting basierenden Firewall. Diese erlaubt nur für vordefinierte Ports Verbindungen und reduziert somit die mögliche Angriffsfläche auf ein Minimum. Da jedoch viele Netze und insbesondere Hochschulnetze deutlich freier und flexibler in ihrer Konfiguration sind, werden Firewalls dort zumeist nur sehr sporadisch und inkonsequent verwendet.

In Abschnitt 2 werden verwandte Ansätze vorgestellt und ihre Einschränkungen diskutiert. Abschnitt 3 erläutert die Erkennung gängiger Port-Knocking-Sequenzen durch Beobachtung des Netzverkehrs konzeptionell. In Abschnitt 4 wird auf die erforderliche Abgrenzung gegenüber typischem Internet-Hintergrundrauschen, z. B. durch verdeckte Portscans, eingegangen. Abschnitt 5 diskutiert anschließend die Möglichkeiten und Grenzen des hier vorgestellten Ansatzes. Abschnitt 6 skizziert schließlich, wie die Erkennungsmechanismen in Controller für Software-defined Networks integriert werden können.

2 Themenverwandte Arbeiten

In der vorliegenden Arbeit wird die Erkennung von versteckten Diensten mit Hilfe von Portscans und der Analyse von Flow Records durchgeführt. Das Interpretieren von Flow Records auf wahrscheinlich ungenutzten Ports wurde von einigen vorherigen Arbeiten bereits betrachtet. Im Folgenden werden diese Arbeiten kurz vorgestellt und deren Unzulänglichkeiten für unseren Anwendungsfall identifiziert.

In [WOK07] stellen Whyte et al. einen Ansatz vor, dessen Zielsetzung dieser Arbeit gleicht. Ihrer Lösung liegt zugrunde, dass ein Port drei Zustände annehmen kann: Entweder ist er vertrauenswürdig, was der Fall ist, wenn der Dienst dahinter bekannt ist; oder er wurde zwar als neuer aktiver Port erkannt, ist aber noch nicht vertrauenswürdig; oder er wird als so genannter Darkport nicht verwendet. Durch diese Einteilung ist es möglich, automatisierte Angriffe und Scans mit einer hohen Wahrscheinlichkeit zu identifizieren. Jedoch bietet der Ansatz keinerlei Möglichkeiten, spezifische Port-Knocking-Sequenzen mitzuprotokollieren; weiterhin werden die zu beobachtenden Systeme alle gleich behandelt, so dass beispielsweise die Backdoor evilshell nicht erkannt werden kann.

Hommes et al. beschäftigen sich in [HSE12] mit der Frage, wie man Port-Knocking-Sequenzen finden kann. Dabei speichern sie alle Flow Records in einer Datenbank und versuchen dann mit Methoden aus dem Data Mining, Ereignisse zu isolieren, die möglichst selten sind (Clustering mit Outlier Detection). Dabei gehen die Autoren davon aus, dass eine Port-Knocking-Sequenz hinreichend speziell ist und dass sie relativ selten verwendet wird. In dem Paper wurde das Testszenario relativ klein gewählt; es steht somit noch der Beweis aus, dass dieser Ansatz auch in größeren Netzen praktisch funktioniert. Weiterhin wurde zwar eine Methode vorgestellt, die Sequenzen zu extrahieren, aber einen weiteren Check, ob danach irgendeine Kommunikation mit dem versteckten Dienst durchgeführt wird, war nicht Teil der Betrachtungen.

Viele weitere Arbeiten, wie beispielsweise [TFVK12], [KSK14], [MS05] oder [DFB+07] beschäftigen sich ebenfalls mit der Auswertung der Kommunikation mit nicht vorhandenen Diensten. Die allgemeine Zielsetzung dieser Arbeiten liegt jedoch schwerpunktmäßig auf der Erkennung von automatischer Ausbreitung von Würmern oder dem Erkennen von Portscans. Mit diesen Arbeiten kann somit nicht unterschieden werden, ob eine Verbindung zu einem Port erfolgreich war oder nicht, da dies bei der Analyse der Wurmausbreitung zumeist nicht von Belang ist.

Aufgrund der dargestellten Einschränkungen wird im Folgenden ein neuer Ansatz vorgestellt, um versteckte Dienste erkennen zu können.

3 Erkennung von Sequenzen

Im Folgenden werden die Funktionalitäten vorgestellt, mit denen versteckte Dienste erkannt werden können. Dazu sind, wie in Abbildung 2 zu erkennen, im Wesentlichen drei Komponenten nötig. Neben einer Komponente zur Aufzeichnung von Flow Records, die üblicherweise durchgehend laufen sollte, und einer Komponente zur Auswertung aller Da-

ten existiert ebenfalls eine Komponente, die regelmäßig Portscans ausführt. Mit Hilfe der Portscans werden reguläre Dienste identifiziert und die damit verbundenen Flow Records aussortiert, um schließlich die somit übrig gebliebenen Flow Records zu analysieren. Diese Komponenten können entweder zentralistisch für ein komplettes Netz oder aber für kleinere Subnetze ausgerichtet sein, wobei jede Komponente unabhängig von der anderen ist, d. h. es kann beispielsweise einen zentralen Flow-Record-Recorder und mehrere verschiedene Portscanner und Auswerteeinheiten geben.

Die hier vorgestellte Erkennung von versteckten Diensten nutzt als Datenbasis zum einen Portscans und zum anderen Flow Records. Die Portscans werden im gewählten Ansatz für zwei verschiedene Zwecke verwendet: Zum einen werden durch einen Portscan alle regulären Dienste identifiziert, da sich ein Netzbetreiber mit Hilfe von Portscans einen Überblick über die in seinem Netz laufenden Dienste verschaffen kann. Werden diese Portscans regelmäßig durchgeführt und miteinander verglichen, so können neue, potentiell nicht erwünschte Dienste aufgespürt werden [vEMH13]. Durch eine Filterfunktion können somit alle Flow Records der regulären Dienste aussortiert werden, was die Menge der zu analysierenden Flow Records signifikant reduziert. Leider ist es unvermeidlich, dass Dienste, die zwischen zwei Portscan-Durchläufen neu installiert oder grundlegend umkonfiguriert werden, in dieser Zeit nicht als reguläre Dienste erkannt werden. Dieses Problem sollte jedoch mit einer Change-Management-Dokumentation, kurzen Zeitintervallen zwischen zwei Portscans oder dem Umstand, dass neue Dienste in der ersten Zeit üblicherweise wenig frequentiert sind, in der Praxis abgefedert werden können.

Zum anderen ist es mit Hilfe von Portscans möglich, mit einer gewissen Wahrscheinlichkeit zu ermitteln, ob es sich bei einem System um ein Server- oder um ein Clientsystem handelt, indem typische Serverdienste ermittelt werden. Dies ermöglicht, unter anderem die evilshell-Backdoor zu erkennen, da es für ein Serversystem üblicherweise hinreichend unwahrscheinlich ist, eine Kommunikation nach außen anzustoßen, wenn es sich nicht um Update-, Zeit-, DNS-Server oder ähnliches handelt, die aber über Ausnahmelisten beherrschbar sind, welche die Ziel-IP-Adressen legitim kontaktierter Server umfassen. Insbesondere ist damit auch die Server-Server-Kommunikation vieler Dienste (z. B. SMTP oder HTTP-Proxy) innerhalb des eigenen Netzes einfach zu entdecken.

Um nun potentielle Port-Knocking-Sequenzen aufzuspüren, werden die Flow Records an einem zentralen Punkt gesammelt und, wie oben bereits erwähnt, mit den letzten Portscanningergebnissen abgeglichen. Dabei werden nur Verbindungen gespeichert, deren Kommunikation mit einem bisher nicht genutzten Port verläuft. Somit werden alle Verbindungen mit bereits bekannten Ports aussortiert. Der im Folgenden beschriebene Ablauf ist in Abbildung 2 dargestellt.

Nun werden die Verbindungen analysiert und in kleine zeitliche Intervalle eingeteilt. Innerhalb dieser Intervalle wird untersucht, ob Verbindungen existieren, bei denen nach mehreren Versuchen mit unterschiedlichen Ports und Protokollen eine erfolgreich aufgebaute Verbindung registriert wurde, wobei die Richtung des erfolgreichen Verbindungsaufbaus irrelevant ist. Dabei muss beachtet werden, dass sich bei den initialen Verbindungsversuchen mindestens einmal der Port oder das Protokoll unterscheiden muss, um beispielsweise häufig in der Praxis anzutreffende mehrmalige Verbindungsversuche wegen Verbindungsfehlern nicht als irreguläres Verhalten zu melden.

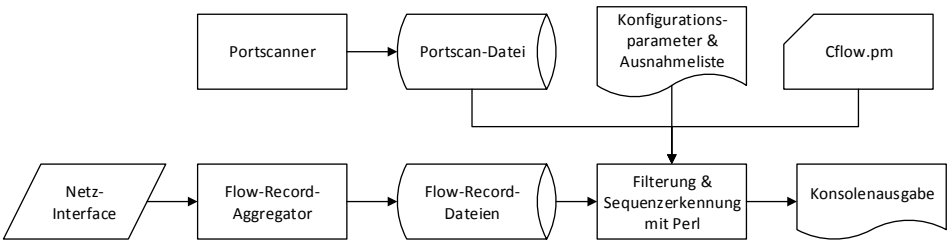

Abbildung 2: Datenerhebung und Auswertung zur Erkennung von Port-Knocking-Sequenzen

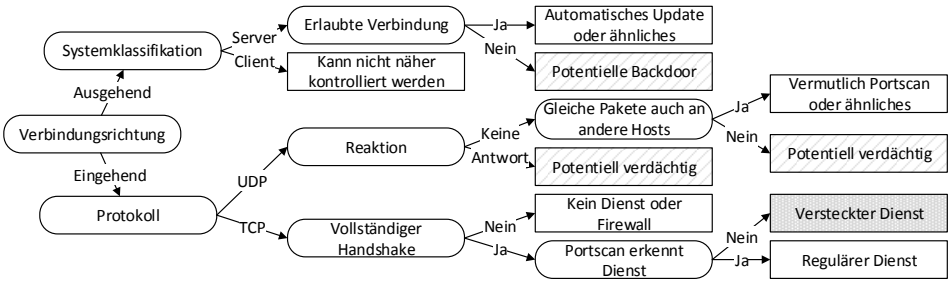

Abbildung 3: Entscheidungsbaum zur Erkennung eines versteckten Dienstes

Da es bei den Malware-Beispielen bislang meist der Fall ist, dass feste Port-Knocking-Sequenzen verwendet werden, können die gefundenen Sequenzen gespeichert werden, was einem Administrator die Möglichkeit verschafft, den restlichen Netztraffic nach genau dieser Signatur abzusuchen, um ggf. weitere Systeme mit den gleichen versteckten Diensten zu identifizieren. Da jedoch ein Angreifer ein solches Verhalten jederzeit mit wenig Aufwand ändern kann oder z. B. nur noch One-Time-Port-Knocking-Sequenzen verwendet, ist dies nur ein optionales Feature und es ist unabhängig von dem restlichen Erkennungsverfahren.

4 Erkennung von Nutztraffic gegenüber Hintergrundrauschen

Unabhängig von der Erkennung der Sequenzen muss ebenfalls erkannt werden, wenn auf einem bisher nicht genutzten Port Traffic zu beobachten ist. Dabei muss man unterscheiden zwischen eingehendem und ausgehendem Traffic. Während der eingehende Verkehr, insbesondere bei Nutzung von TCP, auf ein versteckten Dienst hinweist, ist der ausgehende Verkehr nicht trivial zuzuordnen. Dies ist in Abbildung 3 dargestellt und wird im Folgenden näher erläutert.

Beim eingehenden Verkehr gibt es zwei verschiedene Arten von Transportprotokollen:

verbindungsorientierte und verbindungslose. Bei den verbindungsorientierten Transportprotokollen wird vor dem eigentlichen Senden des Inhalts ein Handshake durchgeführt, der, wenn er vollständig durchgeführt wurde, die Verbindung initialisiert. Durch die Korrelation von Zeitstempeln, Quell- und Ziel-Adressen und der Anzahl der übertragenen Bytes kann von den eigentlich unidirektionalen Flow Records auf den Status des Verbindungsaufbaus geschlossen werden. Ist so ein Handshake zu beobachten, läuft auf der einen Seite mit an Sicherheit grenzender Wahrscheinlichkeit ein Serverdienst. In diesem Fall ist es anzuraten, als Reaktion z. B. einen Portscan samt Diensterkennung auf diesen einen speziellen Port durchzuführen, um zu ermitteln, ob sich nicht inzwischen eine Änderung an der legitimen Portbelegung ergeben hat. Ist durch den Portscan nichts zu ermitteln, so hat man bereits einen versteckten Dienst gefunden, der vom lokalen Systemadministrator näher analysiert werden muss.

Bei den verbindungslosen Transportprotokollen ist eine eindeutige Initialisierung durch ein Handshake nicht vorgesehen, so dass man nicht ad hoc erkennen kann, ob die Nachricht auf der Gegenseite empfangen oder abgewiesen wurde. Um trotzdem zu ermitteln, ob ein versteckter Dienst diese Daten verarbeitet, muss man zum einen beobachten, ob das System durch die empfangenen Informationen eine Reaktion zeigt, d. h. ob von dem System aus Daten – möglicherweise an ein anderes Zielsystem – gesendet werden. Ist dies nicht der Fall, so ist zu überprüfen, ob die Netzpakete ebenfalls an andere Systeme aus dem gleichen oder einem anderen Netzbereich gesendet wurden. Ist dies der Fall, so kann zunächst vereinfachend davon ausgegangen werden, dass die Netzpakete ungezielt versendet wurden und somit kein versteckter Dienst vorhanden ist. Ist dies nicht der Fall, so steigt zwar die Wahrscheinlichkeit, dass ein versteckter Dienst vorhanden ist, aber diese Wahrscheinlichkeit ist so klein, dass eine automatische Reaktion darauf unangemessen wäre und wahrscheinlich mehr Aufwand als Nutzen bringt.

Kann man wiederum ausgehenden Verkehr beobachten, so ist aus Netzmonitoringsicht nur mit näherer Analyse zu beurteilen, ob diese Aktion von einem versteckten Dienst – wie beispielsweise der oben erwähnten Backdoor evilshell – oder von einem Benutzer bzw. legitimen Dienst durchgeführt wird. Dabei ist es wieder hilfreich, auf die Portscanningdaten und die daraus abgeleitete Klassifizierung der Systeme in Server und Clients zurückzugreifen. Dabei ist die Unterscheidung für Clientsysteme nicht durchzuführen, da im Allgemeinen Benutzer die Möglichkeit haben, jegliche Art von Software auszuführen, die dann beliebige Verbindungen aufbauen können.

Hat man es aber mit einem Server zu tun und ist dies kein Gatewayserver bzw. keine Hoppingstation, so kann man davon ausgehen, dass die Hauptaufgabe des Servers darin besteht, Verbindungen von außen anzunehmen und weiterzuverarbeiten bzw. zu festen Kommunikationspartnern eine Verbindung aufzunehmen. Diese ausgehenden Verbindungen können entweder basierend auf Anforderungen vom System bzw. Systemdiensten ausgehen (z. B. Abrufen von Updates oder Zeitsynchronisation mit einem NTP-Server) oder die laufenden Dienste benötigen zur Diensterfüllung eine Kommunikation nach außen. Die Ermittlung, ob diese Kommunikation legitim ist, ist hier zwar möglich, aber der Aufwand, der getrieben werden muss, ist dafür unverhältnismäßig. Für einige passende Verbindungen (z. B. die Authentisierung gegenüber einem LDAP-Dienst) können mit der Zeit Filterlisten eingerichtet werden, aber beispielsweise alle Kommunikationen

zu anderen Mailservern zu bewerten, ist praktisch nicht mit den hier vorgestellten Mitteln durchführbar bzw. würde eine aufwendigere Inhaltsanalyse mit sich bringen, bei der beispielsweise Intrusion Detection Systeme mit Signaturen bekannter Botnet- oder Backdoor-Kommunikationsmuster eingesetzt werden können. Andererseits ist es möglich, anhand der erkannten Dienste Flow-Record-Signaturen zu bilden, welche Kommunikationen üblicherweise durchgeführt werden, da es beispielsweise unwahrscheinlich ist, dass ein standardmäßig konfigurierter NTP-Dienst zu nicht-NTP-Ports Verbindungen aufbaut.

Zusammenfassend kann man daher sagen, dass die Einordnung von eingehenden Verkehr deutlich einfacher zu analysieren ist als ausgehender, bei dem man jedoch bei der Überwachung von Servern mit viel Aufwand bis zu einem gewissen Punkt gelingen kann.

5 Evaluation des gewählten Ansatzes

Um den oben beschriebenen Ansatz evaluieren zu können, wurde ein Testnetz eingerichtet, in dem unterschiedliche Serversysteme installiert wurden. Um den Querschnitt eines Real-World-Netzes zu bekommen, wurden die Systeme jeweils mit unterschiedlichen Windows- bzw. Linuxbetriebssystemen eingerichtet, wobei auf die Systeme willkürlich einige Serverdienste in Betrieb genommen wurden. Der Übergang zum restlichen Rechenzentrumsnetz wurde über ein Gatewaysystem realisiert, das jegliche Kommunikation mittels flowcapture als Flow Records mitprotokolliert hat. Die Auswertung wurde mit einem Perlskript mit der Perl-Bibliothek Cflow.pm realisiert. Schließlich wurde auf einem der Server eine cd00r-Testinstallation eingerichtet.

Während der hier betrachteten Testphase, die sich über etwa einen Monat hinzog, konnten insgesamt 1 116 035 Flow Records gesammelt werden, die durch legitime Nutzungen von Diensten wie SSH, HTTP und Remote Desktop sowie Portscans, Ausnutzen der cd00r-Testinstallation, aber auch durch automatische Updates der Systeme und Synchronisationen untereinander verursacht wurden. Bei der regelmäßigen Analyse der vorliegenden Daten wurden insgesamt 21 potentielle Port-Knocking-Sequenzen erkannt, welche alle fünf Versuche, auf die cd00r-Testinstallation zuzugreifen, enthielten. Es wurden somit erwartungsgemäß alle True-Positives gefunden. Die restlichen erkannten Sequenzen sind alle durch verschiedene Kombinationen von NTP auf Port 123 und NetBIOS auf Port 137 entstanden, stellen also False-Positives dar, die jedoch ausgefiltert werden können.

Es ist somit bei einer produktiven Implementierung darauf zu achten, dass spezifische Protokolle, wie beispielsweise die beobachteten NTP oder NetBIOS, beachtet werden, so dass die Erkennung nicht an diesen aufgrund einer erhöhten False-Positves-Zahl scheitert. Dem kann man begegnen, indem man beispielsweise für den NTP-Zeitdienst eine Ausnahmeregel berücksichtigt, wobei man dabei darauf achten muss, dass man damit einem Angreifer keine Schwachstelle offenbart. Ist das Netz nämlich nicht gegen Spoofing-Angriffe geschützt, so kann ein Angreifer seine Adresse so fälschen, so dass das Analysetool diese Pakete ignoriert. Ist das Netz aber so konfiguriert, dass das Spoofing von eigenen IP-Adressen nicht möglich ist, dann kann man mit Hilfe von IP-Adresse und Portangabe den NTP-Server des eigenen Netzes eindeutig auf eine Ausnahmeliste setzen.

Ein weiterer Parameter, der auf die Analyse einen großen Einfluss hat, ist das betrachtete Zeitfenster, in dem nach zusammenhängenden Sequenzen gesucht wird. In der Testumgebung wurde ein Fenster von 10 Millisekunden gewählt, da die im Internet einfach zugänglichen Backdoor-Implementierungen für Port Knocking bislang kein spezielles Delay vorsehen. Würde allerdings ein Delay verwendet, so kann man die Sequenz nicht mehr erkennen, so dass adaptive Analyseverfahren eingesetzt werden müssten.

Weitere Nachteile sind die Dauer der Analyse und die dadurch entstehende Latenz, die von der Aufzeichnung eines Flow Records bis zur Erkennung der Backdoor auftritt. Bei aktuell in der Praxis eingesetzten Komponenten und Werkzeugen dauert es unter Umständen bis zu 15 Minuten, bis eine beendete Verbindung in der Ausgabedatei des Flow-Record-Aggregators erscheint. Die Analyse der ca. 1 Mio. Flow Records dauert mit der prototypischen Implementierung in etwa eineinhalb Minuten. Durch die Implementierung der Suche nach Sequenzen steigt diese Zeit jedoch exponentiell. Hier könnte eine effizientere und optimierte Implementierung, die eventuell auch durch einen SDN-Controller umgesetzt werden kann, eine deutliche Verbesserung erreichen. In der aktuellen Form ist das Analyseskript nicht dazu geeignet, in einer größeren oder stark belasteten Umbegung eingesetzt zu werden.

6 Portierung auf Software Defined Networks

Software Defined Networks (SDN) wurden entwickelt, um die Flexibilität von Netzen und Netzkomponenten zu erhöhen und somit dynamisch und effizient auf Änderungen reagieren zu können. Zu diesem Zweck wird die logische Aufteilung von Netzkomponenten in Data Plane und Control Plane dahingehend konsequent umgesetzt, dass die Control Plane in Form eines Controllers netzkomponentenübergreifend zentralisiert wird. Die Data Plane übernimmt in SDN das Annehmen und Weiterleiten bzw. Blockieren von Datenpaketen. Dabei ist sie jedoch auf den Controller angewiesen, der die Regeln bzw. Routingtabellen für die Data Plane steuert; diese können auch Layer-4-Angaben enthalten, d. h. es kann z. B. ein TCP-/UDP-Port-spezifisches Routing eingerichtet werden. Wird eine neue Verbindung an der Data Plane empfangen, für die noch keine Regel besteht, weil sie beispielsweise an einen ungewöhnlichen Zielport gerichtet ist, so wird der Controller angefragt, wie das Datenpaket verarbeitet werden soll. Diese Anfrage umfasst auch den Layer-4-Header, so dass der Controller anhand der darin enthaltenen Portangaben u. a. auch Port-Knocking erkennen kann.

Das hier vorgeschlagene Verfahren kann nun ebenfalls als Northbound Interface Application im SDN-Kontext eingesetzt werden, d. h. als Anwendung auf dem SDN-Controller, die über relevanten Datenverkehr informiert wird und optional Einfluss auf die Routingentscheidungen nehmen kann. Dies bringt eine Reihe von Vorteilen: Zum einen ist die oben bereits erwähnte größere Latenz nicht mehr vorhanden, da die Verbindungen direkt analysiert werden können und keine intervallgesteuerte und damit zeitversetzte Analyse mehr erforderlich ist. Weiterhin entfallen alle Konvertierungsschritte, was die Komplexität und damit auch die Fehleranfälligkeit der Implementierung reduziert. Der größte Vorteil ist aber, dass durch eine Anbindung an den Controller eine direkte Reaktion auf beob-

achtete Port-Knocking-Sequenzen möglich ist und nicht erst im Nachhinein festzustellen ist, dass ein Angriff stattgefunden hat. Dies setzt jedoch voraus, dass die Auswertealgorithmen ausreichend effizient für einen Einsatz in Echtzeit geeignet sind, da andernfalls Verzögerungen bei der Zustellung legitimer Datenpakete auftreten können.

7 Fazit

Port Knocking hat sich bewährt, um Dienste wie den SSH-Zugriff auf Server weltweit über das Internet zu ermöglichen, ohne sie den Risiken durch eine direkte Erreichbarkeit durch Angreifer auszusetzen. Das Verfahren funktioniert jedoch so gut, dass selbst die Administratoren nicht mehr einfach erkennen können, welche derart versteckten Dienste in ihren Netzen betrieben werden. Problematisch ist dies insbesondere dann, wenn Systeme kompromittiert wurden und der Angreifer einerseits seine Aktivitäten auf dem System durch Rootkits lokal komplett verbirgt und andererseits eine per Port Knocking geschützte Backdoor einrichtet.

In diesem Beitrag wurde ein Verfahren vorgestellt, mit dem Flow-Record-basierte Aufzeichnungen des Netzverkehrs auf Anzeichen typischer Port-Knocking-Backdoors untersucht werden können. Durch eine Korrelation mit vorhandenem Wissen über legitime Serverdienste im Netz, das beispielsweise mit vorgelagerten Portscans gewonnen wurde, lässt sich der auszuwertende Flow-Record-Datenbestand auf ein beherrschbares Maß reduzieren. Tests einer prototypischen Implementierung in einer Laborumgebung zeigen dennoch, dass noch ein für heuristisches Security-Monitoring nicht unüblicher hoher Anteil an False Positives auftreten kann, die jedoch ausgefiltert werden können, und für die Erkennung komplexerer Port-Knocking-Sequenzen noch wesentlich effizientere Auswertealgorithmen benötigt werden, um nicht nur die derzeit einfach verfügbaren, trivial implementierten Backdoors zuverlässig erkennen zu können.

In der Praxis sind zudem Einschränkungen zu berücksichtigen, die sich u. a. aus der inhärenten Latenz von Flow-Record-Aggregation und -Auswertungen ergeben. Der Ansatz Software-defined Networks, bei dem Controller zum Einsatz kommen, auf denen auch Security-Monitoring-Anwendungen betrieben werden können, bietet eine Reihe interessanter Möglichkeiten, um das hier beschriebene Analyseverfahren eleganter und effektiver implementieren zu können. Im weiteren Verlauf der hier beschriebenen Arbeit sollen deshalb einerseits praktische Erfahrungen in einem Real-World-Netz gesammelt und andererseits verbesserte Algorithmen für SDN-Controller auf Basis von SE-Floodlight implementiert und bewertet werden.

Danksagungen Teile dieser Arbeit wurden durch das Bundesministerium für Bildung und Forschung gefördert (FKZ: 16BP12309). Die Autoren danken den Mitgliedern des Munich Network Management Teams (MNM-Team) für wertvolle Kommentare zu früheren Versionen dieses Artikels. Das MNM-Team ist eine Forschungsgruppe der Münchener Universitäten und des Leibniz-Rechenzentrums der Bayerischen Akademie der Wissenschaften unter der Leitung von Prof. Dr. Dieter Kranzlmüller und Prof. Dr. Heinz-Gerd Hegering.

Literaturverzeichnis

[DFB+07] Guillaume Dewaele, Kensuke Fukuda, Pierre Borgnat, Patrice Abry und Kenjiro Cho. Extracting Hidden Anomalies Using Sketch and Non Gaussian Multiresolution Statistical Detection Procedures. In *Proceedings of the 2007 Workshop on Large Scale Attack Defense*, LSAD '07, Seiten 145–152, New York, NY, USA, 2007. ACM.

[DWH13] Zakir Durumeric, Eric Wustrow und J. Alex Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, Seiten 605–620, Washington, D.C., 2013. USENIX.

[Gra14] Robert David Graham. MASSCAN: Mass IP port scanner. `https://github.com/robertdavidgraham/masscan`, November 2014.

[Gro14] NCC Group. Analysis of the Linux backdoor used in freenode IRC network compromise. `https://www.nccgroup.com/en/blog/2014/10/analysis-of-the-linux-backdoor-used-in-freenode-irc-network-compromise`, Oktober 2014.

[HSE12] Stefan Hommes, Radu State und Thomas Engel. Detecting Stealthy Backdoors with Association Rule Mining. In Robert Bestak, Lukas Kencl, LiErran Li, Joerg Widmer und Hao Yin, Hrsg., *NETWORKING 2012*, Jgg. 7290 of *Lecture Notes in Computer Science*, Seiten 161–171. Springer, Berlin, Heidelberg, 2012.

[KSK14] Manish Khule, Megha Singh und Deepak Kulhare. Netflow Traffic Analyzer For worm detection – A Survey. In *International Journal Of Engineering And Computer Science*, Jgg. 3, Seiten 5441–5446, April 2014.

[Lyo08] Gordon Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure.Com, Sunnyvale, USA, 2008.

[MS05] Joshua McNutt und Markus De Shon. Correlations between quiescent ports in network flows. *Proceedings of the FloCon 2005*, September 2005.

[Nak03] Nawapong Nakjang. A Practical Approach of Stealthy Remote Administration. `http://www.linuxsecurity.com/content/view/117369/49`, August 2003.

[oP00] FX of Phenoelit. cd00r – not listening remote UN*X shell. `http://www.phenoelit.org/stuff/cd00rdescr.html`, Juni 2000.

[Sim08] Simpp. evilshell.c – backdoor remote connect. `http://packetstormsecurity.com/files/69560/evilshell.c.html`, September 2008.

[TFVK12] Florian Tegeler, Xiaoming Fu, Giovanni Vigna und Christopher Kruegel. BotFinder: Finding Bots in Network Traffic Without Deep Packet Inspection. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, Seiten 349–360, New York, NY, USA, 2012. ACM.

[vEMH13] Felix von Eye, Stefan Metzger und Wolfgang Hommel. Dr. Portscan: Ein Werkzeug für die automatisierte Portscan-Auswertung in komplexen Netzinfrastrukturen. In Christian Paulsen, Hrsg., *Sicherheit in vernetzten Systemen: 20. DFN Workshop*, Seiten C-1–C-21, Norderstedt, Deutschland, Januar 2013. Books on Demand.

[VHLT07] Eugene Y Vasserman, Nicholas Hopper, John Laxson und James Tyra. SILENTKNOCK: practical, provably undetectable authentication. In *Computer Security–ESORICS 2007*, Seiten 122–138. Springer, 2007.

[Vin14] Judd Vinet. knockd – a port-knocking server. `http://www.zeroflux.org/projects/knock`, Juni 2014.

[WOK07] David Whyte, Paul C. van Oorschot und Evangelos Kranakis. Tracking Darkports for Network Defense. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, Seiten 161–171, Dezember 2007.