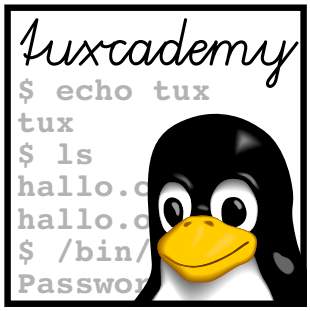
Linux und Sicherheit



tuxcademy – Linux- und Open-Source-Lernunterlagen für alle
www.tuxcademy.org · info@tuxcademy.org

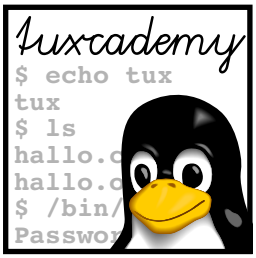
Das tuxcademy-Projekt bietet hochwertige frei verfügbare Schulungsunterlagen zu Linux- und Open-Source-Themen – zum Selbststudium, für Schule, Hochschule, Weiterbildung und Beruf.
Besuchen Sie https://www.tuxcademy.org/! Für Fragen und Anregungen stehen wir Ihnen gerne zur Verfügung.

Linux und Sicherheit

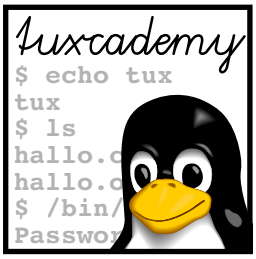
Revision: secu:b18dce0ae917fd16:2014-04-03
nadm:34ccb7a5ca5eb94a:2014-04-03 B
secu:2cd69440d313e762:2013-12-20 1–9
secu:BDN4000Eeo1uMBEgNWoOqd

http://www.tuxcademy.org · info@tuxcademy.org

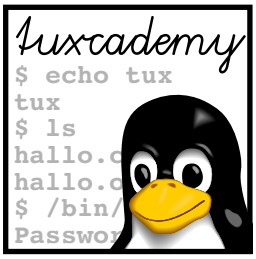
Autoren: Thomas Erker, Stefan Haller, Anselm Lingnau
Technische Redaktion: Anselm Lingnau ⟨*anselm.lingnau@linupfront.de*⟩
Gesetzt in Palatino, Optima und DejaVu Sans Mono



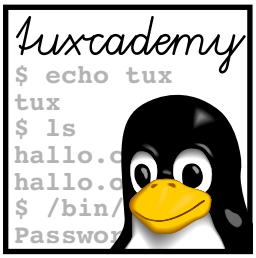
Inhalt



Tabellenverzeichnis



Abbildungsverzeichnis



Vorwort

Diese Schulungsunterlage gibt eine gründliche Einführung in die Sicherheitsadministration von Linux-Systemen. Sie richtet sich an Systemadministratoren mit umfassender Erfahrung mit Linux-Systemen und -netzen und setzt Kenntnisse äquivalent zum LPIC-1-Zertifikat des *Linux Professional Institute* voraus.

Nach einer Erklärung grundlegender Sicherheitsbegriffe und einer Einführung in die Erstellung von Sicherheitskonzepten betrachten wir zunächst die „lokale Sicherheit“ einzelner Linux-Rechner. Wir erklären das Konzept des „Minimalsystems“ und erklären, wie Sie einen Linux-Rechner gegen unbefugte Zugriffe absichern können. Anschließend befassen wir uns mit Sicherheit in TCP/IP-Netzen: Einige besondere Eigenschaften der OpenSSH werden erklärt, bevor wir eine gründliche Einführung in Firewall-Infrastrukturen und die Konfiguration des Linux-Paketfilters „Netfilter“ geben. Kapitel über Angriffserkennung, Sicherheitsüberprüfung und VPN schließen die Unterlage ab.

Diese Schulungsunterlage soll den Kurs möglichst effektiv unterstützen, indem das Kursmaterial in geschlossener, ausführlicher Form zum Mitlesen, Nach- oder Vorarbeiten präsentiert wird. Das Material ist in Kapitel eingeteilt, die jeweils für sich genommen einen Teilaspekt umfassend beschreiben; am Anfang jedes Kapitels sind dessen Lernziele und Voraussetzungen kurz zusammengefasst, am Ende finden sich eine Zusammenfassung und (wo sinnvoll) Angaben zu weiterführender Literatur oder WWW-Seiten mit mehr Informationen.

Kapitel

Lernziele
Voraussetzungen

Zusätzliches Material oder weitere Hintergrundinformationen sind durch das »Glühbirnen«-Sinnbild am Absatzanfang gekennzeichnet. Zuweilen benutzen diese Absätze Aspekte, die eigentlich erst später in der Schulungsunterlage erklärt werden, und bringen das eigentlich gerade Vorgestellte so in einen breiteren Kontext; solche »Glühbirnen«-Absätze sind möglicherweise erst beim zweiten Durcharbeiten der Schulungsunterlage auf dem Wege der Kursnachbereitung voll verständlich.

Absätze mit dem »Warnschild« weisen auf mögliche Probleme oder »gefährliche Stellen« hin, bei denen besondere Vorsicht angebracht ist. Achten Sie auf die scharfen Kurven!

Die meisten Kapitel enthalten auch Übungsaufgaben, die mit dem »Bleistift«-Sinnbild am Absatzanfang gekennzeichnet sind. Die Aufgaben sind numeriert und Musterlösungen für die wichtigsten befinden sich hinten in dieser Schulungsunterlage. Bei jeder Aufgabe ist in eckigen Klammern der Schwierigkeitsgrad angegeben. Aufgaben, die mit einem Ausrufungszeichen (»!«) gekennzeichnet sind, sind besonders empfehlenswert.

Übungsaufgaben

Auszüge aus Konfigurationsdateien, Kommandobeispiele und Beispiele für die Ausgabe des Rechners erscheinen in `Schreibmaschinenschrift`. Bei mehrzeiligen Dialogen zwischen Benutzer und Rechner werden die Benutzereingaben in **`fetter Schreibmaschinenschrift`** angegeben, um Missverständnisse zu vermeiden. Wenn Teile einer Kommandoausgabe ausgelassen wurden, wird das durch »◁◁◁◁◁« kenntlich gemacht. Manchmal sind aus typografischen Gründen Zeilenumbrüche erforderlich, die in der Vorlage auf dem Rechner nicht stehen; diese werden als »▷

◁« dargestellt. Bei Syntaxdarstellungen stehen Wörter in spitzen Klammern (»⟨*Wort*⟩«) für »Variable«, die von Fall zu Fall anders eingesetzt werden können; Material in eckigen Klammern (»[`-f` ⟨*Datei*⟩]«) kann entfallen und ein vertikaler Balken trennt Alternativen (»`-a`|`-b`«).

Wichtige Konzepte
Definitionen

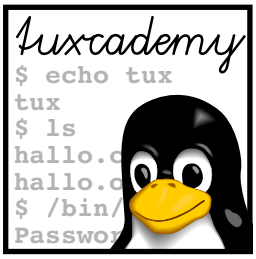
Wichtige Konzepte werden durch »Randnotizen« hervorgehoben; die **Definitionen** wesentlicher Begriffe sind im Text fett gedruckt und erscheinen ebenfalls am Rand.

Verweise auf Literatur und interessante Web-Seiten erscheinen im Text in der Form »[GPL91]« und werden am Ende jedes Kapitels ausführlich angegeben.

Wir sind bemüht, diese Schulungsunterlage möglichst aktuell, vollständig und fehlerfrei zu gestalten. Trotzdem kann es passieren, dass sich Probleme oder Ungenauigkeiten einschleichen. Wenn Sie etwas bemerken, was Sie für verbesserungsfähig halten, dann lassen Sie es uns wissen, etwa indem Sie eine elektronische Nachricht an

```
info@tuxcademy.org
```

schicken. (Zur Vereinfachung geben Sie am besten den Titel der Schulungsunterlage, die auf der Rückseite des Titelblatts enthaltene Revisionsnummer sowie die betreffende(n) Seitenzahl(en) an.) Vielen Dank!



1

Sicherheit: Einführung

Inhalt

Lernziele

- Verstehen, was »Sicherheit« bedeutet
- Einen Überblick über Angriffe und Angreifer bekommen
- Die Schritte zur Aufstellung eines Sicherheitskonzepts kennen
- Informationsquellen zu sicherheitsrelevanten Themen kennen

Vorkenntnisse

- Allgemeine Linux- und Administrations-Kenntnisse

1.1 Was ist Sicherheit?

Wahrscheinlich denken Sie beim Thema »IT-Sicherheit« wie die meisten Anwender an zwielichtige Cracker[1] im Auftrag von KGB oder Mafia, an Viren und Würmer, an ungesicherte WLANs, an Cliff Stolls *Kuckucksei* [Sto93] und an Hollywoodfilme wie *War Games* und *Das Netz*. Zweifellos ist das Internet heutzutage ein gefährlicher Platz, aber IT-Sicherheit besteht nicht nur aus Spionage und Spionageabwehr. Tatsächlich lassen sich drei »Grundpfeiler« der IT-Sicherheit identifizieren:

Vertraulichkeit Viele Sorten von Daten dürfen Unbefugten nicht zugänglich werden. Dies betrifft die vielerorts verarbeiteten »personenbezogenen Daten«, die zumindest in Deutschland auch ausgedehnten gesetzlichen Schutz genießen (anderswo auf der Welt, etwa in den USA, ist das keineswegs so), aber natürlich auch Betriebsgeheimnisse (welcher Hersteller von Dingens würde nicht gerne wissen, was seine Konkurrenten nächstes Jahr auf den Markt bringen wollen?) oder Details der Zugangsmechanismen zu einem Computersystem wie Benutzernamen und Kennwörter. Ein weiterer wichtiger Bereich ist die Kommunikationssicherheit – Vertraulichkeit soll nicht nur für gespeicherte Daten gegeben sein, sondern auch für Kommunikationsinhalte, und es ist oft auch wünschenswert, den Kommunikationspartner eindeutig identifizieren zu können. Hierzu gehört oft auch die »Nichtzurückweisbarkeit« (engl. *non-repudiation*), wo es darum geht, zweifelsfrei beweisen zu können, dass eine bestimmte Kommunikation mit bestimmten Inhalten stattgefunden hat, auch wenn einer der Kommunikationspartner das bestreitet.

Kommunikationssicherheit

Verfügbarkeit Neben der Vertraulichkeit, die sicherstellen soll, dass Unbefugte nicht auf wichtige Daten zugreifen können, ist es wichtig, dafür zu sorgen, dass die rechtmäßigen Benutzer der Daten auch wirklich mit ihnen arbeiten können: Die Daten müssen *verfügbar* sein, und das heißt, dass die Rechnersysteme und die Netze, die sie verbinden, verläßlich funktionieren müssen. Ein robustes Betriebssystem wie Linux kann da schon eine große Hilfe sein, aber zahlreiche Anwendungen erfordern weitere Infrastruktur wie beispielsweise die redundante Auslegung wichtiger Systemkomponenten und die entsprechende Softwarekonfiguration. Dieses Thema – »Hochverfügbarkeit« – ist nicht Gegenstand dieser Schulungsunterlage.

Integrität Der dritte Aspekt der Sicherheit betrachtet im wesentlichen, dass die Daten, die Sie seit gestern abend unbeaufsichtigt gelassen haben, heute morgen noch so sind, wie sie damals waren. Wenn das nicht der Fall ist, können Hardwareschäden schuld sein, aber auch bösartige Cracker löschen heutzutage ja nicht mehr Ihre Festplatte (das wäre ärgerlich, weil Sie sie von Sicherheitskopien rekonstruieren müssen, aber zumeist nicht wirklich existentiell), sondern sie haben herausgefunden, dass es viel wirksamer sein kann, einfach in einer Datenbank oder einer wichtigen Kalkulationstabelle ein paar subtile Fehler einzubauen. Diese Fehler werden möglicherweise erst viel später entdeckt, wenn schon folgenschwere Geschäftsentscheidungen auf der Basis der falschen Daten getroffen wurden. Gerade für börsennotierte Unternehmen kann das große Probleme bedeuten. Im Sinne einer umfassenden IT-Sicherheit ist es also notwendig, die Integrität wichtiger Daten zu sichern – auf der Hardwareebene etwa durch RAID-Systeme

[1] Wir verwenden das Wort »Cracker« im Gegensatz zum volkstümlichen »Hacker«. Ein »Hacker« (ursprünglich ein sehr positiv besetzter Begriff) ist einfach jemand, der sich mit Interesse und Neugier an eine Materie – typischerweise Programmieren – annähert und sich mit ihr beschäftigt, bis er sie nahezu perfekt beherrscht. Zum »Cracker« wird er erst in dem Moment, wo er seine Kenntnisse ausnutzt, um sich beispielsweise Zugang zu Rechnersystemen zu verschaffen, auf denen er nichts zu suchen hat. Umgekehrt ist nicht jeder Cracker automatisch auch ein Hacker – viele Cracker sind dumm wie Bohnenstroh. Wir glauben übrigens auch nicht an die »Hacker-Ethik« des Sich-Nur-Umschauens-Aber-Nichts-Anfassens, da ein Cracker auch durch das bloße Umschauen schon Schaden anrichten kann, oft ohne das überhaupt zu merken.

und regelmäßige Sicherheitskopien, und gegen unerwünschte Manipulation durch geeignete kryptographische Maßnahmen. Natürlich zählen zu den »Daten« auch die Programme, die an der Bearbeitung kritischer Geschäftsdaten beteiligt sind, vom Betriebssystem bis zu den Anwendungsprogrammen.

Überhaupt ist Kryptographie – die Lehre von Methoden zur Verschlüsselung und Entschlüsselung von Daten – ein wichtiger Bestandteil der meisten IT-Sicherheitsverfahren, jedenfalls was Vertraulichkeit und Integrität betrifft. Auf der anderen Seite ist sie kein Allheilmittel, so dass ein Verständnis kryptographischer Verfahren und insbesondere ihrer Grenzen sehr wichtig dafür ist, Aspekte der IT-Sicherheit einschätzen zu können. Eine praktische Einführung in dieses Thema ist [Sch96].

Bedrohungen

Die Probleme der IT-Sicherheit sind nicht grundsätzlich verschieden von den Problemen der Sicherheit im Leben überhaupt. Die Erfahrung zeigt, dass die Bedrohungen in der »digitalen Welt« im Grunde dieselben sind wie die in der »physischen Welt«: Wenn eine »physische« Bank ausgeraubt werden kann, dann auch eine »digitale«. Ein Betrug über das Internet ist nicht sehr verschieden von einem Betrug in der »physischen Welt«. Verträge und Vertragsbruch gibt es überall, genau wie Einbrüche in die Privatsphäre (vom Spanner mit seinem Fernglas über den Cracker, der E-Mails liest, die ihn nichts angehen). Die einzigen Unterschiede ergeben sich daraus, dass Computer und das Internet ein paar Eigenschaften haben, die sich in der physikalischen Welt nicht in derselben Form wiederfinden:

Computer sind schnell Im Gegensatz zu Leuten haben Computer kein Problem damit, eintönige Aufgaben sehr schnell und oft zu wiederholen. Ein Computer kann versuchen, alle Telefonnummern in einem bestimmten Ortsnetz anzurufen, ob sich dahinter ein Modem verbirgt, während das für einen Menschen eine sehr ärgerliche Aufgabe darstellt.

Das Internet hat keine Grenzen Während ein Dieb, der in Ihre Wohnung einbrechen möchte, sich körperlich dorthin begeben muss, ist Ihr Web-Server im Internet prinzipiell von jedem anderen Rechner auf dem Internet aus zugänglich und kann auch von dort angegriffen werden. Die Gruppe der potentiellen Angreifer ist also wesentlich größer als in der wirklichen Welt.

Erfolgreiche Angriffe sprechen sich herum Computer machen es möglich, erfolgreiche Angriffe in vorgekochter Form auch Leuten zugänglich zu machen, die nicht in der Lage wären, sich den entsprechenden Angriff selbst zu erschließen. Auch das vergrößert die Gruppe der potentiellen Angreifer: Während in der wirklichen Welt zum Beispiel das Öffnen von Safes ohne Kenntnis der Kombination etwas ist, das einige Übung und Fachkenntnis erfordert, können in der digitalen Welt auch ansonsten völlig unbedarfte Personen einen vorgefertigten *exploit* verwenden, um eine Sicherheitslücke in einem angreifbaren System auszunutzen.

Inzwischen gibt es sogar Baukästen für Schadsoftware, die es auch technisch weniger versierten Missetätern gestatten, sich neue Viren, Würmer und Trojaner sozusagen »zusammenzuklicken«. Nicht gerade schöne Aussichten für den geplagten Sicherheits-Administrator …

Im Internet gehen die Uhren schneller Die Viren- und Wurmattacken der letzten Monate und Jahre zeigten immer wieder, dass vom ersten Auftreten eines solchen schädlichen Programms bis zu dem Moment, wo Millionen von Rechnern weltweit davon betroffen sind, oft nur sehr kurze Zeitspannen – Stunden oder Tage – liegen. Die »Inkubationszeiten« sind extrem kurz. In der wirklichen Welt breiten Krankheitserreger sich in der Regel wesentlich langsamer aus, so dass mehr Zeit bleibt, um sich Gegenmaßnahmen zu überlegen. Im Internet ist Proaktivität oft erfolgreicher als Reaktivität.

Als Betreiber oder Administrator eines Rechnersystems, das ans Internet angeschlossen ist, handeln Sie unverantwortlich, wenn Sie das Thema »Sicherheit« ignorieren. Selbst wenn Sie selber Sicherheit nicht als persönliches Problem empfinden (typischerweise: »Mir kann ja nichts passieren, ich bin so klein und unauffällig, und außerdem habe ich gute Sicherheitskopien«), so kann es trotzdem sein, dass Sie unfreiwilliger Mittäter werden, etwa wenn Ihr (Windows-)Rechner von einem »Spambot« infiziert oder von einem Cracker in einem verteilten *denial-of-service*-Angriff auf eine wichtige Web-Präsenz mitbenutzt wird. Es liegt an Ihnen, dies möglichst auszuschließen; das Internet ist nur so sicher wie die Summe seiner Teile.

Übungen

1.1 [2] Wie würden Sie versuchen, die Integrität von Datensätzen in einer Datenbank zu sichern?

1.2 Sicherheit als betriebswirtschaftliches Problem

Sicherheit als primär technisches Problem

Eine verbreitete, aber gefährliche Tendenz ist es, Sicherheit als primär technisches Problem anzusehen, das mit technischen Mitteln gelöst werden kann (»Wir installieren einen Firewall, und dann ist unser Netz sicher«). Diese Denkweise wird von Herstellern von »Sicherheitsprodukten« geschürt, die ihren Kunden suggerieren, sie hätten perfekte Lösungen anzubieten, die vom Kunden ohne großen (teuren) Zeitaufwand eingesetzt werden können. Das ist meist ein – mitunter folgenschwerer – Irrtum.

System

Das Problem ist, dass Computersysteme in der Regel nur Teil eines umfassenderen »Systems« darstellen, das beispielsweise auch die daran beteiligten Personen umfasst. Ihr Firewall mag Sie davor bewahren, dass Cracker von außen in Ihr Netz einbrechen und Ihre wichtigen Daten stehlen, aber er hilft Ihnen nicht gegen einen Angestellten, der dazu überredet wurde (mit Geld), dieselben Daten auf einem USB-Schlüsselanhänger am Werksschutz vorbeizuschmuggeln. Ihre Kunden nehmen vielleicht über SSL-verschlüsselte Webseiten mit Ihrem Server Kontakt auf, um sich dort anzumelden, aber sie verraten einer freundlichen Stimme am Telefon ihre Benutzernamen und Kennwörter, wenn die nur überzeugend genug beteuert, in Ihrem Auftrag zu handeln. Und schließlich ist Ihr Firewall-System vielleicht weitestgehend unverwundbar gegen die Angriffe von heute, aber was ist mit den Angriffen von morgen und übermorgen?

Sicherheit: kein fertiges Produkt

Sicherheit ist kein fertiges Produkt, das Sie kaufen können, sondern ein andauernder Prozess – und es gibt keine Standardlösungen »von der Stange«. Wirkungsvolle Sicherheit muss immer Sie als Systembetreiber oder -administrator einbinden, genau wie die Benutzer des Systems, und ist niemals »fertig« in dem Sinne, dass Sie sich nicht mehr weiter darum kümmern müssen.

hundertprozentige Sicherheit

Eine weitere sehr wichtige Beobachtung ist, dass »hundertprozentige Sicherheit« nicht wirklich möglich ist. Je mehr Sie sich an Ihr Ideal von Perfektion annähern, um so teurer wird jede Verbesserung, und zwar exponentiell. Auf die Gefahr hin, einen Gemeinplatz zu zitieren: Sicherheit kostet Geld, und ab einem gewissen Punkt kostet mehr Sicherheit *viel* mehr Geld. Es ist offensichtlich, dass Sie nicht mehr Geld für Sicherheitsmaßnahmen ausgeben sollten, als Sie verlieren würden, wenn der Fall einträte, gegen den diese Maßnahmen Sie schützen sollen. Damit wird Sicherheit aber von einem technischen Problem zu einem Problem der

Versicherungsmathematik

Versicherungsmathematik: Wie wahrscheinlich ist es, dass ein bestimmter Schaden eintritt, und was kostet es Sie, diesen Schaden zu beheben, falls er passiert? Hieraus ergibt sich eine Obergrenze für den Aufwand, den Sie treiben sollten, um diesen Schaden im Vorfeld auszuschließen.

```
Date: Fri, 15 Oct 2004 10:16:05 -0500
Message-Id: <74643946.80880@support@citibank.com>
From: Customer Support <support@citibank.com>
To: Hugo <hugo@example.com>
Subject: NOTE! Citibank account suspend in process

Dear Customer:

Recently there have been a large number of cyber attacks pointing
our database servers. In order to safeguard your account, we require
you to sign on immediately. This process is mandatory, and if you did
not sign on within the nearest time your account may be subject to
temporary suspension.

Please use our secure counter server to indicate
that you have signed on, please click bellow:

  http://221.4.199.31/verification/

Thank you for your prompt attention to this matter and thank you for
using Citibank(R)

Regards,
Citibank(R) Card Department
```

Bild 1.1: »Phishing« nach Kontendaten – ein echter Versuch

1.3 Angriffe

Welchen Angriffen könnten Sie (oder Ihre Rechner) ausgesetzt sein? Hier ein kurzer Überblick:

Destruktive Angriffe »Ein Cracker hat unsere kompletten Daten gelöscht!« – der Alptraum eines jeden Systemadministrators. Oder? Natürlich haben Sie gute Sicherheitskopien, die Sie schnell einspielen können, so dass Ihr System bald wieder zur Verfügung steht (sinnvollerweise nachdem die Sicherheitslücke, über die der Cracker überhaupt Zugang zu Ihrem System bekommen hat, identifiziert und gestopft wurde). Andere Formen von Destruktion sind Viren oder Würmer, die unerkannt auf einem System schlummern, bis sie auf die eine oder andere Weise aktiviert werden und Schaden anrichten – entweder durch plumpes Löschen von Daten oder durch subtile Veränderungen –, oder *denial of service*. Bei letzterem (oft kurz »DoS« genannt) wird nicht direkt Schaden auf dem System angerichtet, indem Daten manipuliert werden, sondern rechtmäßige Benutzer werden daran gehindert, dessen Dienste in Anspruch zu nehmen, etwa indem das System wiederholt neu gestartet oder vom Netz aus unzugänglich gemacht wird oder die Netzanbindung des Systems mit sinnlosen Anfragen überlastet wird, so dass ernstgemeinte Anfragen nicht mehr dazwischen passen. Beim *distributed denial of service* (»DDoS«) verwendet ein Cracker eine Vielzahl – Hunderte oder Tausende – von kompromittierten Rechnern, um beispielsweise einen Webserver so mit Anfragen zu überfluten, dass er keine Chance mehr hat, dem Ansturm Herr zu werden.

Viren

denial of service

Betrug, Bauernfängerei und Identitätsdiebstahl Diese Angriffe richten sich weniger gegen Ihre Rechner denn gegen deren Benutzer: Zwielichtige Angebote sollen sie dazu verleiten, Geldsummen nach Afrika zu überweisen oder ihre Zugangsda-

ten für das Online-Banking in zweifelhafte Web-Formulare einzutragen (Bild 1.1 zeigt ein Beispiel für dieses sogenannte *phishing* – neben dem schlechten Englisch, der Angstmache mit der Kontosperrung und der eigenartigen Message-ID ist der URL mit der IP-Adresse ein hundertprozentiges Indiz für Schindluder). Diese Angriffe können kaum mit technischen Mitteln abgewehrt werden (über den Versuch hinaus, die entsprechenden Nachrichten als solche zu erkennen und zu löschen oder zu kennzeichnen); das einzige, was hilft, ist eine umfassende Aufklärung der Benutzer über die damit verbundenen Gefahren.

Publizitäts-Angriffe Während die vorgenannten Angriffe vor allem von der Motivation »Wie richte ich maximalen Schaden an?« oder »Wie werde ich reich?« beseelt waren, stehen Publizitätsangriffe unter dem Motto »Wie werde ich bekannt/berühmt/berüchtigt«? Dies ist ein völlig anderes Bedrohungsmodell als die anderen, da es nicht darum geht, etwa möglichst viel Geld einzunehmen und es dann möglichst unerkannt zu verprassen, sondern mit einem erfolgreichen »Crack« identifiziert zu werden (und sei es nur unter Pseudonym).

Bei einem Publizitäts-Angriff könnte jemand beispielsweise versuchen, Ihre Web-Seite zu verschandeln (engl. *defacing* – beispielsweise haben Cracker 1996 auf den Web-Seiten der CIA unter anderem das Wort *intelligence* durch *stupidity* ersetzt), oder unter dem Vorwand, »Probleme identifizieren zu wollen«, in Ihr System einbrechen und die Ergebnisse weithin publizieren[2]. Typischerweise ist die daraus resultierende schlechte Presse weitaus destruktiver als aller direkt angerichteter Schaden, insbesondere wenn es um börsennotierte Unternehmen, Internet-Anbieter oder Sicherheits-Beratungsfirmen geht …

Es ist im Übrigen oft überhaupt nicht nötig, sich unbefugt Zugang zu einem fremden Computersystem zu verschaffen, um dort gravierende Sicherheitslücken aufdecken zu können. Im Zeitalter des World Wide Web ist es oft möglich, rein über HTTP-Anfragen ein System dazu zu bringen, vertrauliche Daten frei Haus zu liefern oder im Extremfall Daten zu ändern oder zu löschen. Inkompetente Software-Entwickler zusammen mit liederlich programmierten Werkzeugen (Stichwort: PHP) bieten dafür einen verbreiteten Nährboden, etwa wenn es möglich ist, über ungenügend geprüfte Parameter in URLs oder blauäugig akzeptierte Inhalte von Datenfeldern SQL-Code auf den Server zu transportieren und dort auszuführen. (Genießen Sie dazu `http://xkcd.com/327/`.) Ebenso ist es oft möglich, zum Beispiel durch Javascript-Code in ungeprüften Textfeldern Code in fremden Web-Browsern auszuführen, etwa um deren Sitzungs-Cookies zu stehlen.

Übungen

1.2 [2] »SYN-Flooding« ist eine Form von *denial-of-service*-Angriff. Worum handelt es sich dabei im Detail?

1.3 [2] Wie würden Sie sich (bzw. Ihr System) gegen einen DDoS-Angriff verteidigen?

1.4 [2] Geben Sie einige Kriterien an, die Sie als Sicherheitsbeauftragter Ihren Kollegen empfehlen würden, um *Phishing*-Mails erkennen zu können.

1.4 Angreifer

Die Angreifer lassen sich ebenfalls in mehrere Klassen einteilen:

[2] Hier kommen wir wieder mal in den Bereich haariger ethischer Probleme, die sich am besten mit der Sentenz »Des einen Terrorist ist des anderen Freiheitskämpfer« charakterisieren lassen. Wir sind grundsätzlich sehr dafür, dass Sicherheitsprobleme erkannt und gelöst und, wenn nötig, auch öffentlich angeprangert werden, aber wir sind dagegen, dass so etwas mit illegalen Mitteln passiert. Das unautorisierte »Cracken« fremder Computersysteme ist für uns weit jenseits von dem, was normalerweise noch moralisch akzeptabel ist.

Script Kiddies Die zahlenmäßig größte Klasse von Angreifern stellen die sogenannten *script kiddies* dar. Hierbei handelt es sich um Personen aller Altersstufen, die selbst kein allzu tiefes technisches Verständnis haben, aber vorgekochte *exploits* verwenden, um Sicherheitslücken in verwundbaren Systemen auszunutzen. *Script kiddies* profitieren von billiger Bandbreite und viel Zeit, um systematisch Bereiche von IP-Adressen nach verwundbaren Rechnern abzusuchen; auf kompromittierten Systemen werden gerne Hintertüren und *bots* (kurz für *robots*) installiert. Diese Rechner bilden dann *botnets* zur Verbreitung von Spam oder für DDoS-Angriffe.

Über den Begriff *script kiddie* läßt sich streiten, da er eine Verniedlichung impliziert, die letzten Endes nicht angebracht ist – im Februar 2000 beispielsweise wurde durch DDoS-Angriffe auf die Web-Seiten von Yahoo, E-Trade, CNN und anderen ein Schaden in Millionenhöhe angerichtet, für den als Verantwortlicher ein 15 Jahre alter Kanadier mit dem Spitznamen »mafiaboy« identifiziert und im April 2000 festgenommen wurde. Niemand würde auf die Idee kommen, mit Schusswaffen amoklaufende Schüler als *gun kiddies* zu bezeichnen, auch wenn das grundlegende Problem dasselbe ist: Zugang zu Werkzeugen dafür, großen Schaden anzurichten, und damit verbunden ein absoluter Mangel an Verantwortungsgefühl.

Hinter Botnets stehen inzwischen durchaus handfeste kommerzielle und kriminelle Interessen: Wer ein schönes großes Botnet aufgebaut hat, kann es ganz oder in Stücken an Spammer, Cracker und ähnliche Gestalten vermieten. Botnets haben heutzutage extrem ausgefeilte und schwer zu durchschauende Hierarchiestrukturen von Steuerungsservern (die natürlich auch auf gecrackten PCs laufen), die sich nicht unbedingt leicht bis zu ihren Drahtziehern zurückverfolgen lassen. Die Drahtzieher sitzen sowieso in Ländern wie der Ukraine, wo ihnen mit juristischen Mitteln kaum beizukommen ist, zumal sich in den entsprechenden Kreisen schon praktisch mafiöse Strukturen herausgebildet haben, die nicht nur auf dem Internet, sondern auch in der wirklichen Welt vor wenig zurückschrecken.

Gegen *script kiddies* können Sie sich relativ einfach zur Wehr setzen: Seien Sie möglichst wenig verwundbar gegenüber wohlbekannten Sicherheitslücken (gegen die unbekannten können Sie sowieso recht wenig tun), indem Sie aktuelle Versionen Ihrer Software einsetzen und die entsprechenden Informationsquellen (Abschnitt 1.7) verfolgen. *Script kiddies* haben in der Regel nicht gezielt Sie auf dem Kieker, sondern sind zu vergleichen mit Fahrraddieben am Bahnhof: Wenn zwanzig Fahrräder nebeneinander stehen, wird wahrscheinlich nicht das mit dem fetten Schloss geklaut, sondern das, das gar nicht abgeschlossen ist. Und wenn ein *script kiddie* bei Ihrem Rechner auf Granit beißt, dann macht es halt mit der nächsten IP-Adresse weiter.

Cracker Weitaus gefährlicher als *script kiddies*, wenngleich nicht so zahlreich sind die Cracker, die in der Lage sind, selbst neue Sicherheitslücken zu identifizieren und auszunutzen (die *script kiddies* bekommen ihre *exploits* normalerweise von echten Crackern). Dies nicht nur wegen ihres größeren technischen Könnens, sondern auch, weil sie nicht notwendigerweise nach dem Gießkannenprinzip vorgehen, sondern Installationen gezielt ins Visier nehmen, die ihnen wegen ihrer Ziele mißfallen (Angriffe auf die Web-Server von Abtreibungsgegnern in den USA sind beispielsweise dokumentiert, und im großen IBM-SCO-Linux-Rechtsstreit wurde »die Linux-Szene« mehrfach für DDoS-Angriffe auf `www.sco.com` verantwortlich gemacht – ohne dass das jedoch je belegt werden konnte) oder die anderweitig ihr Interesse erregen. So ist zum Beispiel auch Cracking als Mittel der Industriespionage denkbar.

Weithin überschätzt dagegen wird der »Cyberterrorismus«. In den Nachwehen der islamistischen Anschläge vom 11. September 2001 wurde viel-

fach gemutmaßt, dass Terroristen als nächstes über das Internet Kraftwerke, Staudämme und ähnliche Artefakte manipulieren und für Anschläge einsetzen oder gar das Internet selbst lahmlegen würden. Bisher hat sich nichts dergleichen bewahrheitet: Zum einen scheint den Islamisten eine gut plazierte Ladung Sprengstoff immer noch eindrucksvoller zu sein, und zum anderen brauchen wir erfahrungsgemäß gar keine Terroristen, um das Internet in Unordnung zu bringen – das schaffen Würmer, Viren und Sicherheitslücken auch schon recht effektiv. Die Netze von Atomkraftwerken und ähnlichem sind übrigens nur selten tatsächlich mit dem Internet verbunden; für die Stromausfälle im Nordosten der USA im August 2003 wurde zunächst der `MSBlast`-Wurm verantwortlich gemacht, der etwa zeitgleich im Internet grassierte, obwohl zumindest die offiziellen Verlautbarungen dies später verneinten (genau wie eine Beteiligung der islamistischen Terroristen).

Im Sommer 2010 kursierte der *Stuxnet*-Wurm [STUXNET10], der speziell dafür gedacht war, industrielle Steuerungsanlagen von Siemens anzugreifen (die offenbar, der Leser schaudert, unter Windows laufen). Stuxnet kann solche Rechner ausspionieren und neu programmieren und verbreitet sich über infizierte USB-Sticks. Die genauen Hintergründe sind noch unklar; eine mittlerweile weithin akzeptierte (aber offiziell unbestätigte) Theorie verdächtigt Geheimdienste der USA und Israels, damit Nuklearanlagen im Iran angreifen zu wollen. Tatsache ist, dass Stuxnet eine Komplexität aufweist, die es unwahrscheinlich macht, dass er das Produkt eines einzelnen Hobbyprogrammierers ist, sondern das Programm eher in die Reichweite von gut ausgestatteten Geheimdiensten rückt.

Interne Angreifer Nicht zu unterschätzen sind auch »interne Angreifer«, also Benutzer mit legitimem Zugang zum Netz oder zumindest den Räumlichkeiten. Dies umfasst Angestellte, die sich aus irgendwelchen Gründen auf den Schlips getreten fühlen könnten oder sich ein Zubrot verdienen möchten, genau wie externe Berater oder auch das Wartungs- und Reinigungspersonal. Das Hauptrisiko bei internen Angreifern ist, dass sie sich schon innerhalb des Firewalls befinden und oft auch über detaillierte Systemkenntnisse verfügen, die einem externen Angreifer nicht zugänglich sind. Programmierer können auch »logische Bomben« in der Firmensoftware hinterlassen, die nach einer Kündigung scharf geschaltet werden, oder Hintertüren schaffen, die ihnen auch nach dem Ausscheiden aus der Firma Zugang zum internen Netz geben.

Wie verwundbar Sie gegenüber verärgerten Angestellten sind, ist zunächst eine Frage der Unternehmenskultur – arbeiten Sie daran, dass Ihre Angestellten gar nicht erst verärgert werden, sondern Loyalität gegenüber dem Unternehmen empfinden (oft einfacher gesagt als getan). Externe Berater stöpseln gerne mal ihr Notebook ins Netz und schleppen so einen Virus oder Wurm ein, den Sie ansonsten mühevoll im Firewall ausgefiltert hätten. Wartungstechniker sollten nicht unbeaufsichtigt mit Administratorrechten an wichtigen Systemen arbeiten dürfen, und eventuell sollten Sie auch die Putzkolonne mit Argwohn beäugen: Mehr als nur einmal sind mit Eimern und Mops auch Backup-Bänder aus dem Gebäude getragen worden, auf denen alle interessanten und vertraulichen Daten des Unternehmens standen …

Übungen

1.5 [2] Sie sind verantwortlich für die Netzwerksicherheit eines IP-Providers. Eines Morgens betritt jemand Fremdes unangemeldet Ihr Büro, behauptet, diverse Sicherheitslücken in Ihren Web- und Mailservern gefunden zu haben, und schlägt Ihnen vor, ihn als »Sicherheitsberater« einzustellen. Was halten Sie davon?

1.5 Sicherheitskonzepte

1.5.1 Warum?

Sicherheit ist, wie erwähnt, kein Gebiet, auf dem es Patentlösungen gibt. Deswegen ist es nötig, für eine gegebene Installation ein Sicherheitskonzept aufzustellen, das die genauen Anforderungen für diese Installation wiedergibt. Alle weiteren technischen und administrativen Maßnahmen ergeben sich aus diesem Sicherheitskonzept. Im folgenden besprechen wir die wesentlichen Schritte bei der Erstellung eines Sicherheitskonzepts; für eine weitaus detailliertere Sicht dieses Prozesses empfehlen wir zum Beispiel die *IT-Grundschutz-Kataloge* des Bundesamts für Sicherheit in der Informationstechnik [Inf09].

Sicherheitskonzept

1.5.2 Risikoanalyse

Am Anfang eines Sicherheitskonzepts stehen immer die drei grundlegenden Fragen

- Was versuche ich zu schützen und was ist es mir wert?
- Wogegen muss ich es schützen?
- Wieviel Einsatz, Zeit und Geld möchte ich aufwenden, um angemessenen Schutz zu erreichen?

Diese Fragen sind die Basis einer »Risikoanalyse«, ohne die Sie keine Sicherheitsverbesserung erzielen können (Sie wissen sonst ja nicht, was Sie überhaupt wollen). Wenn zum Beispiel das Risiko eines Stromausfalls besteht und ein andauernder Systembetrieb Ihnen wichtig ist, könnten Sie eine unabhängige Stromversorgung (USV) installieren, um dieses Risiko zu verringern.

Die Risikoanalyse ist ein Prozess, in den Sie am besten erfahrene Benutzer und Führungskräfte aus allen Bereichen Ihrer Organisation einbinden. Stellen Sie Listen Ihrer schützenswerten Objekte und derer Werte auf und identifizieren Sie Bedrohungen. Veranstalten Sie Treffen, in denen diese Listen diskutiert und erweitert werden – ein Vorgang, der nicht nur die Listen verbessert, sondern auch bei allen Anwesenden ein größeres Sicherheitsbewusstsein schaffen sollte.

»Schützenswerte Objekte« umfassen nicht nur konkrete Sachwerte wie Rechner und die darauf gespeicherten Daten, Netzwerkkomponenten wie Router, Kabel und ähnliches, Dokumentation, Distributionsmedien für gekaufte Software, Sicherheitskopien und auch Akten auf Papier, sondern auch »ideelle Werte« wie die Sicherheit und Gesundheit des Personals, Vertraulichkeit personenbezogener Daten, das Bild Ihres Unternehmens in der Öffentlichkeit und gegenüber Ihren Kunden und Zulieferern oder die Verfügbarkeit Ihrer Rechnersysteme. *Es ist wichtig, dass Sie sich bei diesen Betrachtungen nicht nur auf die technischen Aspekte Ihres Rechnernetzes beschränken.*

Die »Bedrohungen« beinhalten nicht nur Naturkatastrophen wie Feuer, Erdbeben, Überschwemmungen oder Explosionen, sondern auch seltene (aber nicht unmögliche) Vorfälle wie dass ein Gebäude einstürzt (denken Sie an das Kölner Stadtarchiv im März 2009) oder Krankheitserreger in der Klimaanlage entdeckt werden, die Sie dazu zwingen, das Gebäude für längere Zeit zu räumen (dem Autor dieser Zeilen passierte das während eines Aufenthalts als Student an einer britischen Universität). Andere Bedrohungen Ihres Systembetriebs umfassen Probleme wie längere Krankheit, plötzlicher Tod oder Kündigung wichtiger Mitarbeiter, Epidemien, durch die mehrere Mitarbeiter ausfallen, Strom-, Wasser-, Telefon- und Internetausfälle für kurze oder lange Zeit, Diebstahl von Systemen, Medien, tragbaren Rechnern, Insolvenz wichtiger Zulieferer, Hardware- und Software-Versagen, Cracker, … die Liste läßt sich fast endlos fortsetzen.

Eine Risikoanalyse sollte kein einmaliges Ereignis bleiben, sondern in periodischen Abständen wiederholt werden, um neue Risiken zu identifizieren und zu berücksichtigen. Dies ist insbesondere dann der Fall, wenn besondere Ereignisse eine Neubetrachtung nötig machen, etwa ein Umzug in eine neues Gebäude oder ein Wechsel des Internet-Providers.

1.5.3 Kosten-Nutzen-Analyse

Liste aller möglicher Risiken

Die Risikoanalyse liefert in der Regel eine kilometerlange Liste aller möglicher Risiken – viel mehr, als Sie sich am Anfang vorgestellt hatten und auch mehr, als Sie in endlicher Zeit mit endlichem Geld ausschließen können. Im nächsten Schritt sollten Sie die Liste der Risiken priorisieren und in Kategorien einteilen – welchen Risiken können Sie technische Maßnahmen entgegensetzen, gegen welche Risiken sollten Sie sich versichern, und welche Risiken sollten Sie einfach ignorieren? Das anerkannte Mittel hierzu ist eine »Kosten-Nutzen-Analyse«, in der Sie jedem Verlust einen Schadensbetrag zuordnen, die Kosten bestimmen, die die Vermeidung dieses Schadens verursachen würde, und die Wahrscheinlichkeit ermitteln, mit der der Schaden eintritt. Anschließend können Sie prüfen, ob die Kosten der Schadensvermeidung den Nutzen der Schadensvermeidung überschreiten.

Kosten eines Schadens

Die Kosten eines Schadens festzulegen ist nicht einfach. Im simpelsten Fall können Sie die Kosten für eine Reparatur oder einen Ersatz des schadhaften Teils ansetzen; im wirklichen Leben müssen wohl noch die Arbeitszeit für die Reparatur, der Schaden durch die Nichtverfügbarkeit, der Rufschaden und ähnliches mit quantifiziert werden. Je mehr Faktoren in die Kostenanalyse einbezogen werden, desto aufwendiger ist der Vorgang, aber desto genauer ist auch das Ergebnis. Üblicherweise genügt es, Schadensklassen zu definieren, etwa beginnend mit »unter €1000« bis hin zu »über €1.000.000« oder »unersetzlich«.

Schadensklassen

Wahrscheinlichkeit

Eher noch schwieriger ist die Analyse der Wahrscheinlichkeit, mit der ein bestimmter Schaden eintritt. Wenn es sich um ein wiederholt auftretendes Problem handelt, dann können Sie Ihre Firmengeschichte heranziehen; ansonsten helfen möglicherweise die Statistiken von Industrieorganisationen, Versicherungen, oder dem Elektrizitätswerk.

Auch Dienstgütezusagen können Indizien liefern – wenn Ihr Internet-Provider Ihnen zum Beispiel »99,9% Verfügbarkeit« verspricht, heißt das andersherum, dass er mit Ausfallzeiten im Bereich von knapp neun Stunden pro Jahr rechnet.

Versuchen Sie herauszufinden, mit welcher Wahrscheinlichkeit ein Schaden pro Jahr auftritt; für seltener oder häufiger auftretende Schäden sollten Sie auch festlegen, mit wie vielen solchen Schäden pro Jahr Sie rechnen (ein schweres Erdbeben findet vielleicht einmal in 100 Jahren statt, woraus eine Wahrscheinlichkeit von 1% pro Jahr resultiert, während schwere Sicherheitslücken in Microsoft Windows monatlich gefunden werden, für eine aggregierte »Wahrscheinlichkeit« von 1200% pro Jahr[3]).

Kosten für Vermeidung

Schließlich müssen Sie die Kosten festlegen, die mit der Vermeidung der jeweiligen Risiken verbunden ist. Neben den »direkten Kosten« können sich auch indirekte Effekte ergeben: Wenn Sie zum Beispiel eine (teure) verbesserte Feuerlöschanlage installieren, kann es sein, dass Ihre Brandversicherung Ihnen mit den Prämien entgegen kommt. Auf der anderen Seite steht das Geld für die Feuerlöschanlage nicht für andere Zwecke im Unternehmen zur Verfügung.

Objekte und Risiken

Am Ende dieses Prozesses sollten Sie über eine lange Liste der schützenswerten Objekte und Risiken verfügen, zusammen mit den Kosten jedes Risikos (der Schaden selbst und dessen Behebung), falls es eintritt, und der Eintrittswahrscheinlichkeit sowie der Kosten für die Risikovermeidung (und im Idealfall einer Wahrscheinlichkeit dafür, dass die Risikovermeidung nicht greift). Sie können nun die Kosten jedes zu erwartenden Schadens mit der Wahrscheinlichkeit

[3] Stochastiker hören hier bitte weg.

für das Auftreten des Schadens multiplizieren und die Liste absteigend nach dem Ergebnis dieser Berechnung sortieren. Die »teuersten« Risiken stehen dann oben. Vergleichen Sie jeden zu erwartenden Schaden mit den Kosten für seine Vermeidung, und Sie erhalten so einen Überblick darüber, welche Schäden es wert sind, vermieden zu werden, und welche Sie lieber in Kauf nehmen sollten. Kümmern Sie sich vor allem um teure wahrscheinliche Schäden und weniger um unwahrscheinliche Schäden mit geringen Kosten.

Bemerkenswerterweise ist es statistisch gesehen wesentlich wahrscheinlicher, dass es bei Ihnen brennt oder Ihnen wichtiges Personal verlorengeht (auf die eine oder andere Art), als dass ein Cracker über das Netz bei Ihnen eindringt, und die daraus resultierenden Schäden sind auch weitaus höher – aber der Crackerangriff wird zumeist als ein viel größeres Problem empfunden[4].

Hier ist ein Beispiel für eine Kosten-Nutzen-Analyse (lose angelehnt an [GSS03, S. 41]): Nehmen wir an, dass der Verlust eines Kennworts von irgendeinem Angestellten im Außendienst dazu führen könnte, dass ein Außenstehender Zugang zu geheimen Firmeninformationen erhält, die €1.000.000 wert sind. Wenn diese Informationen einmal kompromittiert sind, kann der Geheimhaltungsstatus nie zurückgewonnen werden. 40 Angestellte greifen auf Ihr Netz von außen zu, und die Wahrscheinlichkeit, dass einer von ihnen einem Unbefugten Zugang zu seinem Kennwort gibt (etwa indem es über das Internet »erschnüffelt« wird), ist 3% pro Jahr (diese Zahl fällt hier vom Himmel). Das heißt, die Wahrscheinlichkeit, dass mindestens ein Kennwort im nächsten Jahr kompromittiert wird, ist $1 - (1 - 0{,}03)^{40}$, also etwa 70% (schauder), und der zu erwartende Verlust demnach rund €700.000. – Eine mögliche Abhilfe ist die Einführung von Einmalkennwörtern. Nehmen wir an, ein entsprechender S/Key-Rechner kostet €75 pro Benutzer und die dazugehörige Software €10.000 (für Linux sind diese Werte sehr übertrieben), und das System ist fünf Jahre lang zu gebrauchen, dann kostet die Vermeidung des Schadens pro Jahr $(40 \cdot 75 + 10000)/5$ Euro, also €2600 pro Jahr. Es handelt sich offenbar um eine kosteneffektive Lösung.

Restrisiko

Es ist wichtig, hervorzuheben, dass es so gut wie unmöglich ist, ein Restrisiko völlig auszuschließen. Sie können sich mit einer USV gegen plötzliche Stromausfälle schützen, aber wer garantiert Ihnen, dass die USV im Falle eines Falles wirklich funktioniert oder die Putzkolonne sie nicht versehentlich ausstöpselt, weil die Steckdose für den Staubsauger gebraucht wird? Ihre Risikoanalyse sollte versuchen, jeweils auch das Restrisiko zu identifizieren und, wenn möglich, zu quantifizieren.

1.5.4 Sicherheitsziele, Richtlinien und Empfehlungen

Sicherheitsziele

Sicherheitsziele geben vor, welche allgemeinen Erwägungen bei der Umsetzung von Sicherheitsmaßnahmen im Vordergrund stehen sollen. Sinnvollerweise stellen Sie eine einfache Liste von Sicherheitszielen auf (die die Geschäftsleitung verstehen und akzeptieren kann), in der Sie keine konkreten Personen, Rechner und Bedrohungen aufzählen – die Ziele sollen sich möglichst selten ändern. Sie sollten aber allgemein festlegen, welche Daten schützenswert sind, wer für ihren Schutz verantwortlich ist und wer weitere Ziele aufstellen darf. Diese Sicherheitsziele können Sie dann durch Richtlinien und Empfehlungen erweitern, die konkrete Festlegungen treffen. Diese drei Komponenten bilden zusammen ein »Sicherheitskonzept«.

Richtlinien
Empfehlungen

Hier noch einige Tipps für erfolgreiche Sicherheitskonzepte:

[4] Dies entspricht dem Ansatz, Abermillionen auszugeben, um hypothetische Terroranschläge zu verhindern, während man mit demselben Geld, investiert zum Beispiel in Verkehrssicherheit oder medizinische Prävention, weitaus mehr Menschenleben retten könnte.

Benennen Sie »Eigentümer« Für jede Information und jedes Stück Ausrüstung sollte es einen »Eigentümer« geben, der dafür verantwortlich ist, was das Kopieren, Entsorgen, Sichern usw. angeht. In vielen Installationen gibt es wichtige Informationen, für die niemand wirklich zuständig ist, so dass Unklarheit darüber besteht, wer den Zugriff darauf regeln oder die Disposition bestimmen darf. Daten (oder sogar Hardwarekomponenten) verschwinden mitunter für längere Zeit, weil niemand prüft, wo sie sich wirklich befinden.

Seien Sie positiv Ihre Kollegen sind wahrscheinlich nicht begeistert von langen Listen der Form »Tun Sie dies nicht, tun Sie das nicht«. Versuchen Sie, sie zu ermutigen, indem Sie konkrete Anreize geben, aktiv nützliche Dinge zu *tun*, statt schädliche zu *unterlassen*.

Benutzer sind auch Menschen Ein Sicherheitskonzept kann nur dann effektiv sein, wenn die Systembenutzer es mittragen. Wenn Sie versuchen, sie an eine kurze Kette zu legen, und ihnen bei Verfehlungen (die ja in gutem Glauben begangen worden sein oder einfach auf Irrtümern beruhen können) drakonische Sanktionen androhen, dann werden Sie wenig Erfolg dabei haben, die Benutzer zu einer aktiven Unterstützung Ihrer Maßnahmen zu gewinnen. Genausowenig sollten Sie Ihre Benutzer für dumm verkaufen und sich dadurch selbst lächerlich und unglaubwürdig machen.

Ausbildung ist wichtig Sie sollten, wenn irgend möglich, dafür sorgen, dass adäquate Ressourcen für die Aus- und Weiterbildung der Benutzer zur Verfügung stehen. In vielen Systemen ist der Mensch das schwächste Glied (siehe das *Phishing*-Beispiel weiter oben), und entsprechend ausgebildete Benutzer fallen weniger leicht solchen *social-engineering*-Angriffen zum Opfer. Aus- und Weiterbildung sind erst recht für Ihr Systempersonal wichtig. IT-Sicherheit ist ein Feld, das sich sehr schnell weiterentwickelt, und es muss möglich sein, mit diesen Entwicklungen Schritt zu halten. Denken Sie auch daran, dass gute Weiterbildungsmöglichkeiten die Zufriedenheit und Loyalität des Personals steigern und so »Angriffe von innen« weniger wahrscheinlich machen.

Verantwortung und Autorität gehören zusammen Wer im Sicherheitsbereich Verantwortung trägt, sollte auch befugt sein, Maßnahmen durchzusetzen, die die Sicherheit erhalten oder steigern. Dies wird durch »Spafs Ersten Grundsatz der Sicherheits-Administration« illustriert [GSS03]:

> Wenn Sie Verantwortung für Sicherheit haben, aber keine Autorität, um Regeln aufzustellen oder Verstöße zu sanktionieren, dann besteht Ihre Rolle in der Organisation darin, als Sündenbock zu dienen, wenn etwas Großes schief geht.

Klassisch ist die Geschichte des Systemadministrators, der einen Programmierer dabei ertappte, wie er den `root`-Zugang des Personalverwaltungssystems knackte. Der Systemadministrator sperrte augenblicklich den Zugang des Programmierers und beschwerte sich bei dessen Vorgesetztem. Dieser wiederum beschwerte sich bei einem Vorstandsmitglied des Unternehmens über den Administrator und verlangte, dass der Zugang des Programmierers wieder hergestellt werde (seine Arbeit wurde wegen eines Termins benötigt). Der Administrator wurde abgemahnt und drei Monate später entlassen, als jemand in das Personalverwaltungssystem einbrach, für das er verantwortlich war. Der Programmierer wurde befördert. (Auch dieses Beispiel stammt aus [GSS03].)

Sollten Sie sich in einer ähnlichen Situation befinden, dann bewerben Sie sich weg, bevor *etwas Schlimmes passiert.*

Finden Sie eine grundlegende Philosophie In der IT-Sicherheit gibt es zwei grundlegende Philosophien: »Alles, was nicht verboten ist, ist erlaubt« und

»Alles, was nicht erlaubt ist, ist verboten«. Entscheiden Sie sich für eine und seien Sie konsequent.


Die sinnvollere Philosophie ist natürlich die letztere. Es ist immer besser, zuerst alles zu verbieten und dann zu schauen, wer sich beschweren kommt. (Problematisch wird es dann, wenn sich Chefs beschweren kommen, die denken, allein wegen ihrer Position müßten sie alles dürfen. Hier greift wieder »Spafs Erster Grundsatz«.)

Verteidigen Sie sich in der Tiefe Machen Sie nicht bei einer einzigen Verteidigungslinie halt, sondern errichten Sie, wo möglich, mehrfache, unabhängige, redundante Mechanismen. Dazu gehört auch ein Monitoring- oder Alarm-System für den Fall, dass diese Mechanismen nicht funktionieren. Ihr System ist nur so sicher wie die schwächste Komponente.

1.5.5 Audits

Sobald Sie ein Sicherheitskonzept haben, sollten Sie dafür sorgen, dass Ihr System regelmäßig mit dem Sicherheitskonzept verglichen wird. Treten Abweichungen auf, dann können Sie die »Eigentümer« der betreffenden Komponenten dazu bringen, diese zu beheben. Achten Sie dabei darauf, dass es sich um einen kooperativen Prozess handelt, in dem es nicht um Schuldzuweisungen geht, sondern darum, die Systemsicherheit zu erhöhen, indem Probleme identifiziert, Ressourcen besorgt und zugewiesen, Konzepte verbessert und das Sicherheitsbewusstsein erweitert werden.

Audit

Übungen

1.6 [!3] Sie sind verantwortlich für die Sicherheit eines Rechners, der als Dateiserver (mit Samba) für ein Netz mit 30 Arbeitsplätzen dient. Formulieren Sie Richtlinien für die Erstellung von Sicherheitskopien dieses Rechners.


1.7 [!3] Sie sind Systemadministrator für ein Unternehmen mit Standorten in Hamburg (20 Mitarbeiter) und München (15 Mitarbeiter). Zwischen diesen Standorten findet ein reger E-Mail-Austausch über das Internet statt. Betrachten Sie das Risiko, dass Unbefugte von den Inhalten der Nachrichten Kenntnis nehmen und schlagen Sie Maßnahmen vor, um dieses Risiko zu senken. Schätzen Sie die Kosten dieser Maßnahmen ab und beurteilen Sie ihre Kosteneffektivität unter geeigneten Annahmen für den zu erwartenden Schaden, wenn Unbefugte eine vertrauliche E-Mail zu lesen bekommen. Betrachten Sie auch gegebenenfalls notwendige Kosten für die Schulung der Anwender usw.

1.6 Sicherheit und Open-Source-Software

Open-Source-Software im Allgemeinen und Linux im Besonderen werden gerne als »besonders sicher« bezeichnet. Solche Aussagen sind mit Vorsicht zu genießen; die Sicherheit oder Unsicherheit einer Software sind wesentlich enger mit der Kompetenz ihrer Designer und Implementierer verbunden als mit ihrem Entwicklungs- und Vertriebsmodell. Es ist also keineswegs so, dass Open-Source-Software aus prinzipiellen Gründen notwendigerweise sicherer sein muss als proprietäre Software (und Gegenbeispiele wie Sendmail oder Java gibt es zuhauf). Trotzdem lassen sich einige Punkte identifizieren, in denen Open-Source-Software proprietärer Software offensichtlich überlegen ist:

- Ein Hersteller proprietärer Software hat kein Interesse daran, dass Informationen über Sicherheitslücken in seinen Produkten an die Öffentlichkeit kommen. Solche Informationen sind bestenfalls rufschädigend und

schlimmstenfalls zwingen sie ihn zu teuren Patch- und Update-Aktionen, bei denen möglicherweise noch weitere Lücken aufgerissen werden. Ein proprietärer Hersteller wird darum immer versuchen, Sicherheitslücken herunterzuspielen, zu vertuschen, gegen ihre Veröffentlichung vorzugehen (etwa durch einstweilige Verfügungen gegen Redner auf Sicherheits-Konferenzen) und sie heimlich im nächsten (womöglich kostenpflichtigen) Update zu beheben. Open-Source-Projekte dagegen profitieren meist davon, dass Entwickler sich durch Sicherheitslücken »an der Ehre gepackt« fühlen und auf die unvermeidlichen Sicherheitsprobleme sehr zeitnah reagieren, sowie davon, dass eine Sicherheitslücke nicht nur vom ursprünglichen Entwickler des Codes korrigiert werden kann, sondern von jedem, der auf den Quellcode zugreifen kann und über das nötige Know-How verfügt.

- Die Verfügbarkeit des Quellcodes macht es auch möglich, dass unabhängige Experten Open-Source-Software proaktiv auf Sicherheitslücken untersuchen, die dann repariert werden können, bevor tatsächlich *exploits* dafür zur Verfügung stehen. Im Gegensatz dazu können Sie davon ausgehen, dass Hersteller proprietärer Software aktiv dagegen vorgehen werden, dass unabhängige Experten ihre Produkte zu genau unter die Lupe nehmen und ihre Ergebnisse frei veröffentlichen (in den USA machen restriktive Urheberrechtsgesetze wie der *Digital Millennium Copyright Act* (DMCA) das zu einem aussichtsreichen Unterfangen). Manche Datenbankhersteller zum Beispiel betrachten Informationen über Sicherheitsprobleme als »Benchmark- und Leistungsdaten«, deren Veröffentlichung ohne den Segen des Softwareherstellers per Lizenzabkommen untersagt ist und zum Lizenzverlust führen kann. Theoretische Sicherheitslücken interessieren die Hersteller proprietärer Software aus den oben erwähnten Gründen kaum; sie werden in der Regel erst zum Handeln gezwungen, wenn für eine Sicherheitslücke ein *exploit* im Netz kursiert.

  Das oft gehörte Gegenargument behauptet, dass gerade die Verfügbarkeit des Quellcodes Crackern die Gelegenheit gibt, Sicherheitslücken besonders bequem zu finden. Allerdings lehrt die Erfahrung, dass die Cracker offensichtlich keine gravierenden Schwierigkeiten haben, Sicherheitslücken etwa in Microsoft Windows zu lokalisieren, dessen Quellcode *nicht* öffentlich zur Verfügung steht. Tatsächlich hatte Windows bezogen auf den Codeumfang schon deutlich mehr und gravierendere bekannte Sicherheitslücken als Linux. An der Verfügbarkeit des Quellcodes kann es also nicht wirklich liegen.

- »Kerckhoffs' Prinzip« besagt, dass kryptographische Verfahren nur dann als sicher gelten können, wenn ihre Algorithmen weithin bekannt sind und nur der Schlüssel geheim gehalten werden muss. Dieser Grundsatz lässt sich leicht dahingehend ausweiten, dass nur Open-Source-Kryptosoftware vertrauenswürdig sein kann, da Sie (oder die bereits erwähnten unabhängigen Experten) nur bei ihr in der Lage sind, sich zu überzeugen, dass die kryptographischen Algorithmen tatsächlich fehlerfrei implementiert wurden.

Im Falle von Linux sind wir in der glücklichen Lage, dass der größte Teil der sicherheitsrelevanten Software tatsächlich frei bzw. als Open Source zur Verfügung steht. Dies impliziert, wie erwähnt, keine grundsätzliche Überlegenheit von Linux gegenüber proprietären Systemen, was die Sicherheit angeht, aber die Ergebnisse der letzten Jahre sprechen für sich. Linux und die dazugehörige Software – darunter Stützen des Internet wie Apache, BIND und Sendmail – sind beileibe nicht frei von Sicherheitslücken, aber diejenigen, die es gibt, werden erfahrungsgemäß zeitnah behoben.

1.7 Informationsquellen

Ein wichtiger Teil der Arbeit eines Sicherheitsadministrators besteht einfach darin, gut informiert zu sein. Das betrifft nicht nur die Grundlagen (die Sie beispielsweise aus dieser Schulungsunterlage lernen können), sondern auch die Neuigkeiten, die sich fast täglich ergeben: Neue Sicherheitslücken werden entdeckt, diskutiert und repariert, neue interessante Software wird veröffentlicht und vieles mehr. Aus diesem Grund hier eine kurze Zusammenstellung der wichtigsten Informationsquellen rund um Linux und Sicherheit:

Mailinglisten Sie tun gut daran, die Mailingliste Ihrer Distribution zu abonnieren, in der Sicherheits-Updates angekündigt werden. Für die meisten Distributionen gibt es auch Listen, in denen über allgemeine Sicherheitsthemen diskutiert werden kann. Hier die entsprechenden Adressen der gängigsten Distributionen:

Die Mailingliste für Debian-Sicherheitshinweise ist `debian-security-announce@lists.debian.org`; abonnieren können Sie sie unter `http://lists.debian.org/debian-security-announce/`. Allgemeine Diskussionen über Sicherheitsthemen rund um Debian GNU/Linux finden in `debian-security@lists.debian.org` statt (diese Liste ist unmoderiert); zu abonnieren ist diese Liste unter `http://lists.debian.org/debian-security/`.

Für die Enterprise-Produkte von Red Hat gibt es eine Ankündigungsliste namens `enterprise-watch-list@redhat.com`. Eine Anmeldung ist über `http://www.redhat.com/mailman/listinfo/enterprise-watch-list/` möglich. Sicherheitsrelevante Ankündigungen über die Konsumentendistribution von Red Hat, Fedora, sind auf `fedora-announce-list@redhat.com` zu finden (Abonnieren geht sinngemäß). Es gibt auch noch eine `redhat-watch-list@redhat.com` mit Ankündigungen über Red Hat 9.

Informationen über Sicherheits-Updates für die openSUSE-Distribution erscheinen auf `opensuse-security-announce@lists.opensuse.org`. Allgemeine Diskussionen können unter `opensuse-security@lists.opensuse.org` geführt werden. Abonnieren können Sie diese Mailinglisten über `http://lists.opensuse.org/`. Für den SUSE Linux Enterprise Server gibt es keine öffentlich zugängliche Security-Mailingliste; Ihnen als Lizenzkunde wird Novell Näheres verraten.

Auf allen diesen Listen erscheinen Nachrichten auf Englisch.

Wenn Sie sich nicht nur für Ihre Distribution interessieren, sondern über Sicherheitsthemen insgesamt informiert werden wollen (und Zeit haben), ist die Mailingliste der Wahl »BUGTRAQ«, zu finden unter `http://www.securityfocus.com/archive`. Dort werden viele Sicherheitslücken zum ersten Mal bekannt gegeben, im Detail diskutiert, und es werden auch *exploits* veröffentlicht (die Sie als verantwortlicher Administrator natürlich nur verwenden würden, um die Sicherheit Ihrer eigenen Systeme zu prüfen).

Web-Seiten Diverse Web-Seiten beschäftigen sich mit dem Thema »Sicherheit«. Aus diesem Grund können wir hier nur einige aufzählen, es gibt viel mehr:

LWN.net Als Einstieg nützlich und auch von allgemeinem Interesse ist LWN.net (`http://lwn.net`), ehemals *Linux Weekly News*. Hier erscheinen in einem tickerartigen Format Mitteilungen über wichtige Sicherheitslücken sowie eine tägliche Zusammenfassung von Sicherheits-Updates der gängigen Distributionen. Ebenfalls sehr empfehlenswert ist die wöchentliche Ausgabe, die jeweils donnerstags erscheint und eine feste Rubrik “Security” enthält, in der die Ankündigungen der Woche (Lücken und Updates) zusammengefasst und auch Hintergrundinformationen zu ausgewählten Themen gegeben werden. (Die

wöchentliche Ausgabe ist in der ersten Woche nach ihrem Erscheinen nur Abonnenten zugänglich; ein LWN.net-Abonnement ist ab $3,50 pro Monat erhältlich.) Unter `http://lwn.net/security` stehen die neuesten Ankündigungen von Sicherheitslücken und Updates sowie eine Themenliste der wöchentlichen Ausgaben im Archiv zur Verfügung.

Common Vulnerabilities and Exposures Da diverse Distributionen und Betriebssystemplattformen in weiten Teilen dieselbe Software verwenden, ist es nützlich, Ankündigungen verschiedener Hersteller korrelieren zu können. Ferner ist es sinnvoll, Sicherheitslücken eindeutig zu identifizieren und zu katalogisieren, einfach um zu wissen, wovon man redet. Diese Aufgabe erfüllt der "*Common Vulnerabilities and Exposures*"-Index (CVE), der im Web unter `http://www.cve.mitre.org/` eingesehen werden kann. Jede gefundene Sicherheitslücke bekommt eine CVE-Nummer, mit der sie dann eindeutig benannt ist und auf die man sich in Diskussionen und Ankündigungen beziehen kann. Viele Softwareprodukte im Sicherheitsbereich, etwa Sicherheits-Scanner, verwenden CVE-Nummern, um auf gefundene Lücken hinzuweisen.

SecurityFocus Unter `http://www.securityfocus.com` steht eine weitere Seite zur Verfügung, die allgemeine Sicherheitsinformationen anbietet.

LinuxSecurity Diese Seite unter `http://www.linuxsecurity.com/` versteht sich als »zentrale Stimme für Linux- und Open-Source-Sicherheits-Neuigkeiten«. Noch eine tickerartige Seite.

Zusammenfassung

- Die wesentlichen Aspekte von IT-Sicherheit sind Vertraulichkeit, Verfügbarkeit und Integrität.
- Zwischen den Bedrohungen der »wirklichen« und »digitalen« Welt bestehen keine qualitativen Unterschiede. Lediglich die größere Geschwindigkeit des Computers für wiederholte Aufgaben sowie die verbesserte Kommunikation über das Internet werfen neue Probleme auf.
- Niemand ist klein und unwichtig genug, um Sicherheitsprobleme ignorieren zu können.
- Sicherheit ist ein betriebswirtschaftliches Problem, kein technisches; absolute Sicherheit ist nicht bezahlbar.
- Sicherheitskonzepte bilden den administrativen Rahmen für Sicherheitsmaßnahmen; sie definieren Ziele und Verantwortlichkeiten und stellen Richtlinien und Empfehlungen auf, mit deren Hilfe die Ziele erreicht werden sollen.
- Open-Source-Software ist nicht per se »sicherer« als proprietäre Software, unterscheidet sich jedoch in der Philosophie des Umgangs mit Sicherheitslücken und erlaubt eine unabhängige Prüfung des Quellcodes.
- Zu Sicherheitsthemen stehen diverse Mailinglisten und Web-Seiten zur Verfügung, wo Sie sich umfassender informieren können.

Literaturverzeichnis

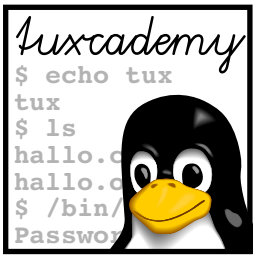
GSS03 Simson Garfinkel, Gene Spafford, Alan Schwartz. *Practical Unix & Internet Security*. Sebastopol, CA: O'Reilly & Associates, 2003, 3. Auflage.
http://www.oreilly.com/catalog/puis3/

Inf09 Bundesamt für Sicherheit in der Informationstechnik. *IT-Grundschutz-Kataloge*. Köln: Bundesanzeiger-Verlag, 2009. ISBN 978-3-88784-915-3. Kostenfrei auf DVD über das BSI zu beziehen oder herunterzuladen.
http://www.bsi.bund.de/DE/Themen/ITGrundschutz/itgrundschutz_node.html

Sch96 Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1996, 2. Auflage. ISBN 0-471-12845-7, 0-471-11709-9. Deutsch als *Angewandte Kryptographie* (Addison-Wesley).

Sto93 Clifford Stoll. *Kuckucksei – Die Jagd auf die deutschen Hacker, die das Pentagon knackten*. W. Krueger, 1993. ISBN 978-3810518620. Derzeit nur gebraucht erhältlich.

STUXNET10 Aleksandr Matrosov, Eugene Rodionov, David Harley, et al. »Stuxnet under the microscope«, September 2010.
http://www.eset.com/resources/white-papers/Stuxnet_Under_the_Microscope.pdf



2

Lokale Sicherheit

Inhalt

Lernziele

- Einen Linux-Rechner gegen unbefugten Zugriff sichern können
- Die Wichtigkeit von Minimalsystemen einschätzen können
- Sicherheitseigenschaften des BIOS und der Bootlader LILO und GRUB kennen

Vorkenntnisse

- Linux-Administrationskenntnisse
- PC-Hardwarekenntnisse sind von Vorteil

2.1 Physische Sicherheit

2.1.1 Physische Sicherheit – warum?

»Physische Sicherheit ist das, was stattfindet, bevor Sie Kommandos auf der Tastatur eintippen« [GSS03] – alle baulichen und anderen »nicht computertechnischen« Maßnahmen, die den Zugriff zu Ihren Rechnern sichern helfen. Schließlich nützt das beste Firewall-System nichts, wenn ein Angreifer sich als Wartungstechniker getarnt ins Haus schmuggelt und den kompletten Server mitnimmt. Sie sollten also in einem sicherheitsrelevanten Umfeld die Frage nach der physischen Sicherheit nicht ignorieren.

Bei Schutzmaßnahmen aus dem Bereich der physischen Sicherheit ist eine Risikoanalyse besonders wichtig, da Gegenmaßnahmen oft ziemlich teuer sein können. Sie sollten also sorgfältig abwägen, welche Risiken Sie durch geeignete Maßnahmen vorbeugend ausschließen bzw. minimieren wollen (etwa Schäden durch einen Stromausfall durch eine geeignete Notstromversorgung), bei welchen Sie den Schaden minimieren wollen, sobald er entsteht (etwa durch ein Reserve-Rechenzentrum in einer anderen Stadt) und welche Sie als Restrisiko hinzunehmen bereit sind. [GSS03] weist darauf hin, dass keine wie auch immer geartete physische Sicherheitsmaßnahme die Mieter des World Trade Center am 11. September 2001 vor dem Zusammenbruch des Gebäudes hätte schützen können. Auch dürften die wenigsten Installationen einem konzertierten (para-)militärischen Angriff widerstehen. Das ist aber kein Grund, physische Sicherheit völlig zu ignorieren – ein Fall wie der des 11. September ist ein dringendes Argument für ein Reserve-Rechenzentrum anderswo (oder zumindest für aktuelle anderswo aufbewahrte Sicherheitskopien).

2.1.2 Planung

Wie bei der Aufstellung von Sicherheitskonzepten im allgemeinen sollten Sie bei der Planung physischer Sicherheit damit beginnen, Ihre Ziele und den *status quo* zu katalogisieren, um einen Überblick darüber zu bekommen, wo Sie stehen und welche Schritte (möglicherweise) noch notwendig sind. Wie sehen Ihre Rechner, Router usw. aus und wo stehen sie? Wie wertvoll sind die darauf gespeicherten Informationen? Wie ist die Grenze zwischen Ihrem sicherheitsrelevanten Bereich und dem »Rest der Welt« ausgestaltet und welche Lücken enthält sie? Gegen welche Gefahren wollen (und können) Sie sich absichern? Was würde das kosten? Zumindest sollten Sie sich die folgenden fünf grundlegenden Fragen stellen [GSS03, Kap. 8]:

1. Hat jemand außer Ihnen jemals physischen Zugriff zu Ihren Rechnern?
2. Was würde passieren, wenn diese Person einen Wutanfall bekommt und mit einem Hammer auf Ihre Rechner losgeht?
3. Was würde passieren, wenn ein Angestellter Ihres größten Wettbewerbers unbemerkt ins Gebäude käme?
4. Wenn es in Ihrem Gebäude brennte und die Rechner unbenutzbar würden, würde das Ihre Organisation lähmen oder zugrunde richten?
5. Wenn Ihrem System irgendeine Katastrophe zustieße, was würden Sie den Benutzern sagen?

Sie sollten, wenn möglich, einen Katastrophenplan für den Fall aufstellen, dass Ihre Rechner Ihnen durch Feuer, Diebstahl oder technischen Defekt abhanden kommen, *und Sie sollten diesen Plan auch ausprobieren*, etwa indem Sie sich vergleichbare Hardware ausleihen und Ihre Sicherheitskopien darauf zu installieren versuchen.

2.1.3 Risiken

Hier sind einige der wichtigsten »physischen« Risiken, denen Ihre Rechner ausgesetzt sein können:

Essen und Trinken Die meistignorierte Sicherheitsregel sagt: »Keine Speisen und Getränke in der Nähe eines Rechners«. Sie können eine Rechnertastatur kaum effektiver außer Gefecht setzen als indem Sie eine Kaffeetasse oder ein Colaglas darüber ausleeren.

Die Hersteller von Industrie-PCs und Notebook-Rechnern kontern dieses Problem inzwischen durch abgedichtete Tastaturen oder solche mit einem Ablaufloch, durch das die Flüssigkeit austreten kann, ohne Schaden anzurichten.

Unbefugter Zugang Sie sollten Unbefugte daran hindern, sich Zugang zu Ihren Rechnern zu verschaffen. Achten Sie auf doppelte Böden und abgehängte Decken, große Ventilationsschächte und Glaswände. (Denken Sie an Filme wie *Sneakers* oder *Ocean's Eleven*.)

Feuer Achten Sie auf gute Feuerlöschausrüstung in der Nähe Ihrer Rechner und darauf, dass das Systempersonal damit auch umgehen kann. Feuerlöscher müssen regelmäßig gewartet werden. Sorgen Sie dafür, dass das Systempersonal Zugang zu einem Telefon hat, am besten einem, das nicht über Ihre Telefonanlage geschaltet ist, sondern direkt am Netz der Telefonfirma hängt. – Oft überstehen Rechner das eigentliche Feuer, werden dann aber vom Löschwasser zerstört. Wenn Sie eine Sprinkleranlage haben, dann sorgen Sie dafür, dass die Rechner automatisch den Strom abgeschaltet bekommen, bevor die Sprinkleranlage angeht (und achten Sie auch auf Ihre USV).

Rauch Brandrauch und auch Tabakrauch sind nicht gut für Computer. Rauch, der beim Brand von Rechnern entsteht, ist mitunter sogar giftig und kann andere Rechner beschädigen. Verwenden Sie Rauchmelder und verbieten Sie das Rauchen in der Nähe Ihrer Rechner – und denken Sie auch an doppelte Böden und abgehängte Decken, über die Rauch und Gase sich ausbreiten können.

Wasser Eine Überschwemmung im Rechnerraum (durch Naturkatastrophen oder die Bemühungen der Feuerwehr) ist nicht notwendigerweise eine totale Katastrophe, solange die Rechner zum betreffenden Zeitpunkt nicht in Betrieb waren. Lassen Sie die Rechner *gründlich* trocknen und investieren Sie gegebenenfalls in eine professionelle Reinigung. – Moderne Festplatten sind nicht luftdicht abgeschlossen, sondern haben Luftlöcher, durch die Wasser eintreten kann; Löschwasser ist Trinkwasser und damit leidlich sauber, aber Flußwasser ist schmutzig. Es kann einfacher sein, die Platten der betroffenen Systeme in neue Rechner einzusetzen (nach dem Trocknen!) und die Daten darauf *sofort* auf neue Platten umzukopieren, oder gleich ein Speziallabor zu beauftragen, das die Platten in einer hochreinen Umgebung auseinandernimmt, sauber macht und kopiert. Als vorbeugende Maßnahme bieten sich Feuchtigkeitsmelder an, die Sie am besten in Bodennähe anbringen. Im Idealfall haben Sie *zwei* Feuchtigkeitsmelder in unterschiedlicher Höhe (unterhalb Ihrer Rechner); der niedrigere Melder sollte einen Alarm auslösen, der höhere automatisch den Strom für die Rechner abschalten. *Diese Maßnahme kann Leben retten.*

Stromversorgung Je nach Ihrer Umgebung können Ihre Rechnersysteme über Spannungsspitzen im Netzstrom in Mitleidenschaft gezogen werden. Diese entstehen durch schwere Geräte, Motoren oder andere Computer in der Nähe (ein Staubsauger reicht mitunter!). Sicherheitsmaßnahmen umfassen hier zum Beispiel die Verwendung eines getrennten Stromnetzes für Computer

(und Zubehör) und andere technische Geräte wie Staubsauger, Wasserkocher und ähnliches und/oder die Installation von Filtern in der Stromversorgung. Statische Elektrizität kann elektronische Komponenten beschädigen; verwenden Sie antistatische Teppichböden oder entsprechende Putzmittel.

Blitzschlag Blitzschläge in der Nähe können zu richtig großen Spannungsspitzen führen (so dass die gängigen Filter dagegen nicht mehr helfen). Die Telefonleitungen sind heute zum größten Teil gegen Blitzschlag gesichert, aber Sie sollten es sich dringend verkneifen, im Freien Kupferkabel für Ethernet o. ä. ohne metallene Kabelkanäle zu verlegen. Blitzschläge können auch beträchtliche Magnetfelder erzeugen und dadurch magnetische Medien (Backup-Bänder) löschen.

Vandalismus und Terrorismus Computer sind leicht zu zerstören, etwa aus Rache oder ideologischen Gründen, oder einfach weil es Spaß macht (?). Neben den Computern selbst sind auch die Netzwerkkabel dankenswerte Ziele dafür. Verwenden Sie Kabelkanäle (und wenn Sie es ernst meinen, Kabelkanäle aus Stahl, die unter Überdruck stehen). Gegen terroristische Anschläge im Stil des 11. September können Sie sich, wie erwähnt, kaum wirklich schützen; hier ist eine effektive Strategie für Sicherheitskopien »anderswo« die einzige Möglichkeit, entstehenden Schaden zu minimieren. Auf der anderen Seite ist die Eintrittswahrscheinlichkeit dafür unter normalen Umständen extrem gering.

2.1.4 Diebstahl

Diebstahl ist letzten Endes auch »nur« eines der physischen Risiken, aber vermutlich dasjenige, das im wirklichen Leben am häufigsten auftritt. Rechnerdiebstahl ist ärgerlich, möglicherweise teuer und, je nachdem, welche Daten auf dem Rechner gespeichert sind, eventuell katastrophal vernichtend. Dabei können einige einfache Maßnahmen das Diebstahlrisiko für Rechner erheblich senken.

Besonders diebstahlgefährdet sind natürlich tragbare Rechner – Notebooks und PDAs. Wachsamkeit ist hier erste Priorität; Mitarbeiter mit solchen Systemen sollten zu besonderer Aufmerksamkeit angehalten werden. Der Wiederverkaufswert von gestohlenen Notebooks läßt sich mindern, indem man spezielle Aufkleber anbringt, die sich nicht wieder entfernen lassen, oder (Firmen-)Name und Adresse ins Gehäuse eingravieren läßt. Handelsüblich sind auch spezielle Schlösser, mit denen Sie ein Notebook zumindest temporär an einem schweren und unbeweglichen Objekt befestigen können, und die sich nur mit einem Schlüssel öffnen lassen oder indem man das Gehäuse des Rechners beschädigt (was den Wiederverkaufswert extrem reduziert).

Beachten Sie, dass »Notebook-Napping« nicht notwendigerweise auf Wiederverkauf ausgerichtet sein muss. Mitunter wird ein Notebook-Benutzer gezieltes Opfer einer Attacke auf die Vertraulichkeit der Daten auf seinem Rechner. In solchen Fällen ist es sinnvoll, die Dateisysteme auf dem Notebook zu verschlüsseln und sie so vor unbefugter Kenntnisnahme zu sichern.

Oftmals lassen gestresste Manager in ihrer Hektik ihren Rechner im Restaurant oder der First-Class-Lounge stehen (und geben das möglicherweise in ihrer Firma als »Diebstahl« an, weil beklaut zu werden nicht ganz so ehrenrührig ist wie vergeßlich zu sein). Auch für diesen Fall ist es wichtig, den rechtmäßigen Eigentümer gut sichtbar und unentfernbar auf dem Gerät kenntlich zu machen, damit das ehrliche Personal den Computer leichter zurückgeben kann als behalten.

2.1.5 Alte Medien

Es ist bemerkenswert, was für interessante Daten man auf den Platten von gebrauchten Computern finden kann – entweder einfach so oder indem man sich

die Platte etwas genauer anschaut. Die meisten Betriebssysteme löschen Dateien nämlich nicht wirklich, sondern kennzeichnen sie nur als »gelöscht und später mal zu überschreiben«, insbesondere um Benutzern die Gelegenheit zu geben, die Dateien notfalls dem Orkus zu entreißen. Mit geeigneten Werkzeugen ist es dann leicht, die ursprünglichen Dateien wieder herzustellen.

Dasselbe gilt übrigens auch für das Neupartitionieren oder gar Formatieren der Festplatte oder sogar das Überschreiben der Platte mit Nullbytes. Hier ist es sukzessive schwieriger, an die gelöschten Daten heranzukommen, wenn auch nicht unmöglich (das Lesen einer komplett überschriebenen Platte bedarf der Dienste eines Speziallabors). Ob das für Sie und Ihre Institution ein Risiko darstellt, mit dem Sie rechnen müssen, müssen Sie wissen; wir empfehlen Ihnen trotzdem, Platten, die Sie aussondern, entweder physikalisch zu zerstören (ein Vorgang, der am besten einen großen Hammer involviert und nur einen besonders entschlossenen Angreifer nicht davon abschrecken dürfte, die Ergebnisse Ihrer Bemühungen ins Speziallabor zu tragen) oder mit einem speziellen Löschprogramm zu behandeln, etwa `wipe`. Hier werden die Daten nicht einfach gelöscht, sondern mit einer geeigneten Sequenz von Bitmustern überschrieben, die auch die Speziallabors frustrieren dürfte [Gut96]. Das ganze dauert natürlich sehr lange.

Löschprogramm

`wipe` kann prinzipiell einzelne Dateien löschen, aber bei den heute unter Linux üblichen journalbasierten Dateisystemen hilft das präzise gar nicht; die scheinbar gelöschten Daten stehen in der Regel anderswo auf der Platte als die Resultate der Überschreiboperationen. Verwenden Sie `wipe`, um eine *komplette* Platte zu löschen, bevor Sie sie oder den Rechner weg geben, etwa indem Sie den Rechner von einem Rettungssystem (etwa einer Knoppix-CD oder ähnlichem) starten und ein Kommando wie

`wipe` und Journal-Dateisysteme

```
# wipe /dev/sda
```

eingeben. (Lassen Sie das Kommando am besten über Nacht laufen.)

Bedenken Sie auch, dass moderne Festplatten in der Lage sind, Zugriffe auf schadhafte Blöcke transparent auf Blöcke einer »Geheimreserve« umzulenken. Das heißt, dass möglicherweise vertrauliche Daten in leicht beschädigter Form aus den ursprünglichen Blöcken zu lesen sind (jedenfalls für das Speziallabor), Sie diese Blöcke aber nicht mehr zum Löschen zu fassen kriegen.

schadhafte Blöcke

Schließlich sollten wir noch erwähnen, dass die in modernen Rechnern verbauten SSDs Daten auch bunt im Speicher verteilen, um einzelne Blöcke nicht übermäßig oft zu beschreiben und damit zu sehr abzunutzen. Außerdem enthalten gerade Hochleistungs-SSDs normalerweise mehr Speicherplatz, als sie für ihre nominale Kapazität brauchen, um die Lebensdauer zu erhöhen. Hier hilft im Zweifelsfall nur totale physische Zerstörung (siehe unten).

Geheimdienste wie die NSA verlassen sich nicht auf Löschprogramme; Platten, auf denen entsprechend klassifizierte Daten standen, werden zu kleinen Krümeln verarbeitet und die Krümel erhitzt, bis alle magnetische Information darauf verlorengeht. Das Resultat der Prozedur unterliegt immer noch der Geheimhaltung. Magnetbänder werden verbrannt.

Für CD-ROMs und ähnliche optische Medien ist »totale physische Zerstörung« die Methode der Wahl. Schneiden Sie die CD mit einer Schere durch, oder kleben Sie Paketband auf die Oberseite und reißen Sie es schwungvoll ab, so dass die reflektierende Schicht daran hängen bleibt. Für bessere Resultate packen Sie sie in eine Plastiktüte, tun diese in eine gefaltete Zeitung und bearbeiten Sie sie mit einem Hammer auf einem Betonboden. Dies macht Ihre (Ex-)Daten sicher vor

CD-ROMs

allen unbefugten Lesern mit Ausnahme der Jungs (und Mädels) von den Anstalten mit dreibuchstabigen Namen. Um diese zu frustrieren, verwenden Sie eine Schleifmaschine und schleifen Sie die Oberseite der CD-ROM ab, bis Sie auf das durchsichtige Material kommen.

Bitte vernichten Sie alte CD-ROMs *nicht* in der Mikrowelle. Der Kunststoff gibt gesundheitsschädliche Gase ab, deren Reste ihren Weg in Speisen finden könnten, die später in dem Ofen zubereitet werden. Auch wenn die Lichteffekte nett aussehen …

2.2 Minimalsysteme

Eine alte Volksweisheit besagt: »Die sichersten Komponenten eines Computersystems sind die, die es nicht gibt.« Als Administrator eines Systems, für das entsprechende Sicherheitsanforderungen gelten, sollten Sie also Sorge tragen, nur solche Softwarepakete zu installieren, die für die Aufgaben des Systems erforderlich sind. Auf einem Rechner, der als Mail-Server fungiert, muss es nicht notwendigerweise auch einen Web-Server geben, denn ein Cracker könnte eine Sicherheitslücke im Web-Server ausnutzen, um die wesentliche Funktionalität des Rechners – die Mail-Zustellung – zu kompromittieren. Auf einem sicherheitskritischen System sollten Sie zum Beispiel auch keine C-Entwicklungsumgebung installieren – auch wenn Cracker durchaus in der Lage sein dürften, ihre Programme anderswo zu übersetzen, muss man es ihnen nicht einfacher machen als nötig.

Leider unterstützen die wenigsten gängigen Linux-Distributionen eine Installation als wirkliches »Minimalsystem«. Ihre Standardinstallationen, selbst die angeblichen »Minimalsysteme«, sind viel zu fett und enthalten mengenweise Material, das auf einem sicherheitskritischen Server nicht benötigt wird. Es ist schwierig, so ein System auf einen Stand »abzumagern«, der unseren Ansprüchen an ein Minimalsystem; oft ist es erfolgversprechender, mit einem schlankeren System, etwa Debian GNU/Linux, anzufangen, und Dinge *hinzuzufügen* (wobei man auch einem Debian-System noch etwas Speck abtrainieren kann).

Wie groß der Unterschied zwischen einem »Minimalsystem« und einer aktuellen Linux-Distribution sein kann, können Sie daran sehen, dass es durchaus funktionelle Linux-Distributionen gibt, die ein Basis-Linux nebst Webserver auf ein paar Disketten (!) unterbringen. Zur Ehrenrettung der »großen« Distributionen muss man natürlich sagen, dass ein solches Miniatur-Linux natürlich Software verwendet, die etwas abseits des »Mainstream« ist, etwa Busybox statt Bash und GNU-`coreutils`. Während es sich technisch also noch um ein »Linux« handelt, können sich in Benutzung und Administration also durchaus Unterschiede ergeben. Naja, niemand hat gesagt, dass Sicherheit nichts kostet …

Mitunter kann es sinnvoll sein, auch den Linux-Kernel auf einem sicherheitskritischen System auf das Nötigste abzuspecken, das für die Hardware des Rechners nötig ist. Jede unbenutzt herumliegende Softwarekomponente kann Sicherheitslücken enthalten, die einen Ansatzpunkt dafür bilden, das ganze System zu kompromittieren. Viele Quellen raten deshalb dazu, Kernels für sicherheitskritische Systeme statisch (also ohne Module) zu übersetzen und nur die Treiber für die wirklich vorhandenen Geräte einzubinden. Auch bei Dateisystemen, Netzunterstützung und so weiter können Sie in der Regel das Allermeiste fortlassen.

Ein anderes Argument, das für den Verzicht auf Kernel-Module ins Feld geführt wird, ist die Existenz von sogenannten *root kits* auf Kernel-Modul-Basis. Ein *root kit* ist ein Softwarepaket, das ein Cracker auf einem kompromittierten System installiert, um sich später wieder bequem Zugang verschaffen zu können, und es ist klar, dass so etwas sich gründlich verstecken muss – Sie als lokaler Administrator sollen ja keine verdächtigen Einträge in der Ausgabe von Programmen wie `ls`, `ps` oder `netstat` sehen, die

Ihre Aufmerksamkeit auf die Dateien, Prozesse und Netz-Sockets des *root kits* lenken. Plumpe *root kits* kommen daher mit modifizierten Versionen von `ls` & Co., die diejenigen Teile der Ausgabe, die sich mit Bestandteilen des *root kits* befassen, einfach unterschlagen. Da Sie aber vielleicht Software laufen haben, die die Integrität von wichtigen Systemprogrammen wie `ls` prüft, damit man Ihnen keine »trojanischen Pferde« unterschiebt (Kapitel 7 stellt einige Methoden dafür vor), sorgen modernere *root kits* dafür, dass der *Kernel* die entsprechenden Informationen schon auf der Systemaufruf-Ebene verschwinden läßt – Sie können weiterhin die Originalversionen von `ls` & Co. verwenden, werden aber trotzdem getäuscht! Diese Funktionalität wird meistens über ein Modul in den zu kompromittierenden Kernel eingeschleppt, deshalb liegt es nahe, einen Kernel zu verwenden, der überhaupt keine Module laden kann.

Heute gibt es auch *root kits*, die sich direkt über das `/dev/kmem`-Gerät in den Kernel hineinpatchen können, selbst wenn dieser überhaupt keine Module unterstützt. Da es für `/dev/kmem` im wesentlichen keine Anwendung außer *root kits* zu geben scheint – für die ursprünglichen Zwecke, etwa Kernel-Debugging, gibt es inzwischen geschicktere Methoden –, ist mit seinem baldigen Verschwinden aus dem Standard-Kernel zu rechnen. (*"There are no forward compatibility guarantees for root kit authors."* [Cor05].)

Übungen

2.1 [2] Wie minimal ist »minimal«? Sollte ein minimaler Web-Server zum Beispiel einen Secure-Shell-Daemon für Fernzugriff enthalten, oder widerspricht das der These, dass keine unnötigen Komponenten installiert sein sollten? Welche Maßnahmen könnten Sie ergreifen, um einen sicheren Fernzugriff auf einen Server-Rechner zu realisieren?

2.2 [4] Versuchen Sie, einen »Minimal-Kernel« für Ihren Rechner zu konfigurieren, zu übersetzen und zu installieren. Dieser Kernel sollte keine Module unterstützen, sondern direkt alles Nötige enthalten, um das System als Server zu betreiben. Verzichten Sie auf alle überflüssigen Gerätetreiber oder Funktionen – beispielsweise könnten Sie »Video for Linux«, die 3D-Grafik oder die (V)FAT-Dateisystemunterstützung ausklammern.

2.3 [5] Auf welche Größe können Sie die »Minimalinstallation« einer Standarddistribution wie openSUSE, Debian GNU/Linux oder Fedora abmagern? Was können Sie alles (bequem) weglassen? Welche Maßnahmen würden Sie einem Distributionshersteller vorschlagen, um die Erstellung von Minimalsystemen zu erleichtern?

2.3 Den Bootvorgang sichern

2.3.1 Bootvorgang und BIOS

Wer physischen Zugang zu Ihrem Rechner hat, kann versuchen, ihn neu zu starten, um statt `init` eine Shell zu starten, oder eine mitgebrachte Knoppix-CD (oder ähnliches) verwenden, um sich Administratorzugriff zu verschaffen und das `root`-Kennwort zu löschen. Grundsätzlich läßt sich das nicht völlig ausschließen, aber Sie können es einem Angreifer sehr schwer machen, sich bis zu einer `root`-Shell auf Ihrem System vorzukämpfen. Natürlich kann ein Angreifer immer die Platten ausbauen und in einen Rechner einsetzen, auf dem er schon `root` ist; in letzter Konsequenz hilft gegen einen Angreifer mit physischem Systemzugriff also nur die Verschlüsselung der Dateisysteme – und selbst das funktioniert nur, wenn der Angreifer den Rechner nicht im eingeschalteten Zustand (mit Schlüsseln für die Dateisysteme im RAM) antrifft.

Es gibt Methoden, RAM-Chips dazu zu bringen, ihren Inhalt auch ohne Strom für eine Weile zu konservieren (Stichwort »Kältespray«) und trickreiche Gadgets, mit denen (typischerweise) die Polizei einen beschlagnahmten Rechner »laufend« abtransportieren kann. Versprechen Sie sich also nicht zu viel von Festplattenverschlüsselung für Ihren Server.

BIOS

Erste Verteidigungslinie gegen unbefugte Angreifer ist das BIOS des Rechners. Hier sollten Sie dafür sorgen, dass der Rechner standardmäßig statt von Floppy (!?), USB-Stick oder einem optischem Medium von der Platte bootet, um zu verhindern, dass jemand eine bootfähige CD einsetzt, um sich Wartungszugriff zu verschaffen. Sie sollten die BIOS-Einstellungen über ein Kennwort gegen unbefugtes Ändern sichern und im Idealfall ein BIOS verwenden, das kein wohlbekanntes Master-Kennwort hat, das das von Ihnen gesetzte überstimmen kann.

2.4 Bootlader-Sicherheit

2.4.1 Grundsätzliches

Nach dem BIOS kommt der Bootlader an die Reihe. Uneingeschränkter Zugriff auf die heute üblichen Bootlader ist unter dem Strich gleichbedeutend mit uneingeschränktem Zugriff auf den Rechner selbst, da ein Angreifer das System im Einbenutzermodus hochfahren oder das Init-System (und alle damit verbundenen Kennwortabfragen) komplett umgehen und in eine Shell booten kann. Möglicherweise könnte er den Rechner sogar über das Netz einen ganz anderen Kernel und/oder ein ganz anderes Root-Dateisystem laden lassen.

Kennwortschutz

Ähnlich wie das BIOS erlaubt auch der Bootlader normalerweise, den Zugriff auf Nicht-Standard-Optionen unter einen Kennwortschutz zu stellen. Die Details hängen vom Bootlader ab.

2.4.2 GRUB 2

Der heute von den meisten gängigen Distributionen eingesetzte Bootlader ist GRUB 2. In der Standardeinstellung erlaubt er einem Benutzer an der Konsole eine ungeahnte Flexibilität bei der Konfiguration – was für die Fehlersuche ein großer Segen ist, für den gestressten Sicherheitsadministrator hingegen eher ein Fluch. So können Sie nicht nur über das GRUB-Menü den Bootvorgang komplett umkrempeln, sondern auch bestimmen, wie der Linux-Kernel aufgerufen wird,

GRUB-Shell

oder gar die »GRUB-Shell« starten, die Ihnen Zugriff auf (fast) beliebige Dateien erlaubt.

GRUB-Menü

Normalerweise zeigt der Rechner beim Systemstart für eine mehr oder weniger kurze Zeit das »GRUB-Menü« an, in dem verschiedene Startoptionen (außer dem normalen Start etwa ein Start in einem »abgesicherten Modus«, in dem fehleranfällige Kernel-Optionen ausgeschaltet sind, oder ältere Kernel-Versionen, um bei Upgrades eine Rückfallposition zu haben) angeboten werden. Sie können mit den Pfeiltasten eine der Optionen wählen und mit ↩ starten – oder mit e die GRUB-Konfiguration für diese Option zum Ändern aufrufen.

Interessant ist in der Konfiguration zunächst die Zeile, die mit `linux` anfängt – sie gibt an, wie der Linux-Kernel aufgerufen wird. Das Erste, was Sie probieren

`single`

können, ist, ans Ende der Zeile das Schlüsselwort »`single`« anzuhängen. Wenn Sie danach den Bootvorgang starten (mit @@FIXME@@), bootet Linux direkt in den Einbenutzermodus, wo möglicherweise keine Kennwortabfrage stattfindet.

`init=/bin/sh`

Viele heutige Distributionen fragen auch für den Einbenutzermodus das `root`-Kennwort ab. Dann müssen Sie statt »`single`« eben »`init=/bin/sh`« verwenden. Diese Option führt statt des normalen `init`-Prozesses eine Shell aus, so dass Sie direkt und ohne Kennwortabfrage eine als `root` laufende Shell bekommen sollten. Allerdings wird die normale Sequenz von Aktionen beim Systemstart nicht ausgeführt; Sie sollten also nicht damit rechnen,

dass der Rechner seinen Namen kennt, mit dem LAN verbunden ist oder auf Dateisysteme außer dem Root-Dateisystem zugreifen kann (und selbst auf dieses möglicherweise nur lesend). Trotzdem könnten Sie zum Beispiel das `root`-Kennwort löschen oder ein zweites Benutzerkonto mit der UID 0 anlegen, das Ihnen dann nach einem Neustart zur Verfügung steht – oder Sie kopieren einfach wichtige Dateien (etwa `/etc/shadow`) auf einen USB-Stick.

GRUB absichern

Wenn Sie GRUB absichern wollen, sollte es Ihnen also um drei verschiedene Dinge gehen:

1. Zunächst sollten Sie unprivilegierte Benutzer daran hindern, die verschiedenen Startoptionen beliebig ändern zu können.
2. Außerdem sollen unprivilegierte Benutzer nicht die GRUB-Shell aufrufen dürfen.
3. Schließlich könnte es sein, dass unprivilegierte Benutzer nur Zugang zu bestimmten vordefinierten Startoptionen haben sollen. Zum Beispiel könnten Sie aus Bequemlichkeit einen Eintrag im GRUB-Menü haben, der den Rechner von einem USB-Stick oder optischen Medium startet, aber dieser Eintrag könnte Systemadministratoren vorbehalten bleiben, damit normale Benutzer nicht auf dumme Gedanken kommen (wie auf dem Firmen-Notebook die neueste SteamOS-DVD auszuprobieren).

`superusers`

Die ersten beiden Punkte bekommen Sie in den Griff, indem Sie in der GRUB-Variablen `superusers` eine Liste derjenigen Benutzer angeben, die administrativen Zugriff auf GRUB haben sollen. Nur diese Benutzer dürfen Startoptionen ändern oder die GRUB-Shell aufrufen:

```
set superusers="admin"
```

Bitte machen Sie das nicht direkt in der normalerweise automatisch erzeugten `/boot/grub/grub.cnf`, wo diese Einstellung höchstwahrscheinlich baldigst wieder überschrieben wird; bei der weithin üblichen aufgestückelten GRUB-Konfiguration ist der richtige Platz dafür die Datei `/etc/grub.d/40_custom`.

mehrere Benutzer

Sie können in `superusers` ohne Weiteres mehrere Benutzer benennen, indem Sie deren Namen durch Leerzeichen, Kommas, Semikolons, Pipe-Symbole oder & voneinander trennen.

Rollennamen

Trotzdem sind Sie mit einem »Rollennamen« höchstwahrscheinlich besser beraten. Wenn Sie in `superusers` die Namen aller Ihrer Administratoren aufzählen, dann werden Sie sich spätestens dann verfluchen, wenn ein neuer Kollege dazukommt (oder ein alter das Unternehmen verläßt) und Sie auf allen 100 Servern, 387 Arbeitsplatz-PCs und 500 Außendienst-Notebooks die GRUB-Konfiguration anpassen müssen. (Das Kennwort zu ändern wird schon mühselig genug.)

Wir benutzen hier absichtlich `admin` und nicht `root`, um zu unterstreichen, dass der GRUB-Benutzer `admin` mit dem Linux-Benutzer `root` nichts zu tun hat. Sie dürfen sich den Namen aber natürlich frei aussuchen.

Kennwort

Nachdem Sie `admin` zum »Superuser« erklärt haben, müssen Sie ihm noch ein Kennwort zuordnen. Dafür haben Sie zwei Möglichkeiten: Das GRUB-Kommando `password` erlaubt eine Vergabe im Klartext:

```
set superusers="admin"
password admin geheim
```

Das ist bequem, bedeutet aber, dass Sie auf die Dateien `/etc/grub.d/40_custom` und `/boot/grub/grub.cnf` aufpassen müssen wie ein Schießhund. Besser ist es, ein verschlüsseltes Kennwort zu verwenden, das das Kommando `grub-mkpasswd-pbkdf2` Ihnen liefert:

```
$ grub-mkpasswd-pbkdf2
Passwort eingeben: geheim
Passwort erneut eingeben: geheim
PBKDF2-Prüfsumme Ihres Passworts ist grub.pbkdf2.sha512.10000.F2AC▷
 ◁ A01A085FE38CE49A6C1BF2C43486107EC109EA3031E1DA536318F30AF426CB7E▷
 ◁ 800C95639E95BBE7BE20300C8AB64F02F1BD746AE533D213F7C916724580.5A5▷
 ◁ EF6C8DF59BF0E2721C4AA662EA3EB11C9DEFE9EE6DCEBC94CA1AB782108149CB▷
 ◁ DE4C0DAF130BAB7576E355A8DF7F44B2A68015B5716D98FC5FC6874D53C56
```

Diesen Rattenschwanz müssen Sie (in einer Zeile) in die GRUB-Konfiguration eintragen:

```
set superusers="admin"
password_pbkdf2 admin grub.pkdf2.sha512.10000.F2AC◁◁◁◁◁
```

Anschließend erzeugt

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

die eigentliche GRUB-Konfiguration neu.

Bei Debian GNU/Linux heißt das offizielle Kommando `update-grub`, aber das ist auch nur ein Zwei-Zeilen-Shellskript, das `grub-mkconfig` aufruft.

Mit `password` und `password_pbkdf2` können Sie auch Benutzer definieren und mit Kennwörtern versehen, die nicht in `superusers` aufgezählt werden. Diese »normalen« Benutzer dürfen dann zum Beispiel Einträge im GRUB-Menü aufrufen, die für sie speziell freigeschaltet wurden, aber für die »Allgemeinheit« nicht zugänglich sind.

benannte Benutzer

Wenn die `superusers`-Variable gesetzt ist, dürfen nur die darin benannten Benutzer überhaupt andere Einträge aus dem GRUB-Menü auswählen. Mit »`--users`« in einem Menüeintrag können Sie bestimmte Benutzer benennen, die auf diesen Eintrag Zugriff haben sollen (ohne dass sie an ihm herumbasteln dürfen); mit »`--unrestricted`« können Sie Einträge kennzeichnen, die allen Benutzern ohne Kennwortangabe zur Verfügung stehen sollen:

```
menuentry "Nur für Administratoren" --users "" {
  ◁◁◁◁◁
}
menuentry "Für alice (und Administratoren)" --users alice {
  ◁◁◁◁◁
}
menuentry "Für alle zugänglich" --unrestricted {
  ◁◁◁◁◁
}
```

2.4.3 GRUB Legacy

Die (Prä-2.0-)Version von GRUB befindet sich auch noch im Umlauf und ist möglicherweise auf älteren Rechnern installiert. Das Meiste, was wir über GRUB 2 gesagt haben, gilt auch für GRUB Legacy – mit der wesentlichen Einschränkung,

kein Benutzerkonzept

dass GRUB Legacy kein Benutzerkonzept hat. Statt dessen werden in der GRUB-Konfigurationsdatei (meistens `/boot/grub/menu.lst` direkt Kennwörter vergeben, etwa mit

```
password geheim                                          Klartext …
password --md5 $1$o9oKo/$gVUEu3Uif6N.byF9fQMp70          Verschlüsselt!
```

Das verschlüsselte Kennwort besorgen Sie sich mit dem Kommando `grub-md5-crypt`:

```
# grub-md5-crypt
Password: GeHeIm
Retype password: GeHeIm
$1$o9oKo/$gVUEu3Uif6N.byF9fQMp70
```

Dabei müssen Sie es wieder von Hand in die Datei `/boot/grub/menu.lst` kopieren.

`password`-Kommando

Wenn das `password`-Kommando im »allgemeinen« Teil der Konfigurationsdatei steht (also vor der ersten »`title`«-Zeile, die einen Menüeintrag definiert), muss es für jegliche Interaktion mit GRUB jenseits der Auswahl einer vorgegebenen Menüoption eingegeben werden. Steht es innerhalb eines Menüeintrags, also nach einer »`title`«-Zeile, dann gilt es nur für den betreffenden Menüeintrag. Auf diese Weise können Sie unterschiedliche Menüeinträge mit unterschiedlichen Kennwörtern schützen.

Mit GRUB 2 geht das nur über den Umweg, für diese Menüeinträge verschiedene Benutzer zu definieren.

Wenn Sie im allgemeinen Teil ein Kennwort gesetzt haben, können Sie innerhalb einer Konfiguration auch das Schlüsselwort `lock` verwenden. Damit ist es notwendig, das allgemeine Kennwort eingegeben zu haben, bevor die betreffende Konfiguration gestartet werden kann.

Auf einer `password`-Zeile können Sie auch noch einen Dateinamen angeben. Dieser Dateiname benennt eine neue GRUB-Konfigurationsdatei, die nach der Eingabe des korrekten Kennworts gelesen wird und weitere Konfigurationen enthalten kann.

2.4.4 LILO

Der Bootlader LILO wird von den gängigen Distributionen bei Neuinstallationen nicht mehr verwendet, allerdings ist es möglich, dass Sie ein uraltes System warten müssen oder eines, das zwar aktualisiert wurde, bei dem der Bootlader aber nicht umgestellt wurde.

Die oben angesprochenen Abkürzungen in den Einbenutzermodus oder die völlig ungeprüfte `root`-Shell erreichen Sie bei LILO, indem Sie den Namen einer Konfiguration gefolgt von `single` oder `init=/bin/sh` angeben. Da die LILO-Standardkonfiguration der meisten Systeme `linux` heißt, ist das keine große Hürde:

```
linux single                    Einbenutzermodus
linux init=/bin/sh              Kein Init-System, nur die Shell
```

Wie bei GRUB besteht die Abhilfe gegen diese billigen Tricks darin, es normalen Benutzern zu verbieten, dass sie auf der LILO-Eingabezeile clevere Parameter angeben. Dies erreichen Sie am leichtesten dadurch, dass Sie die LILO-Konfiguration durch ein Kennwort schützen. Wenn Sie in der Datei `/etc/lilo.conf` (*Achtung:* Name distributionsabhängig) im »globalen Bereich« (also vor der ersten `image=`-Zeile) etwas angeben wie

```
password=geheim
restricted
```

dann bootet LILO ausschließlich die Standardkonfiguration. Um andere Konfigurationen auswählen oder beim Booten gar beliebige Parameter übergeben zu können, müssen Sie (oder wer auch immer) das Kennwort eingeben.

Achten Sie darauf, dass legitime Benutzer nicht das Kennwort ausspähen können, das unverschlüsselt in der LILO-Konfigurationsdatei steht. Sorgen Sie zum Beispiel dafür, dass nur `root` diese Datei lesen kann.

Sie können auch die Kommandozeilen einzelner Konfigurationen vor Manipulation schützen, indem Sie die `password`- und die `restricted`-Zeile in die jeweilige Konfiguration aufnehmen.

Um die Latte noch ein kleines bisschen höher zu legen, können Sie die LILO-Konfigurationsdatei »unveränderlich« machen, indem Sie das entsprechende Dateiattribut setzen:

```
# chattr +i /etc/lilo.conf
```

Vor jeder Änderung muss das Attribut erst wieder zurückgesetzt werden. (Einen cleveren Cracker hält sowas natürlich nicht lange auf, aber es gibt ja jede Menge unclevere Cracker.)

Übungen

2.4 [!2] Sichern Sie den Zugriff auf den Bootlader auf Ihrem System über ein Kennwort. Vergewissern Sie sich, dass Konfigurationsänderungen nur vorgenommen werden können, wenn vorher das korrekte Kennwort eingegeben wurde. (Für GRUB-Anwender: Testen Sie den Unterschied zwischen `--users` und `--unrestricted` in einer Konfiguration; GRUB-Legacy-Anwender sollten den Unterschied zwischen `lock` und `password` ausprobieren.)

2.5 [2] Richten Sie (sofern nicht schon vorhanden) einen Menüeintrag im Bootlader ein, der es erlaubt, den Rechner von einem optischen Medium zu starten. Stellen Sie sicher, dass nur ein Systemadministrator diesen Eintrag benutzen darf.

Kommandos in diesem Kapitel

`grub-md5-crypt` Bestimmt MD5-verschlüsselte Kennwörter für GRUB Legacy
`grub-md5-crypt`(8) 28

`grub-mkconfig` Erzeugt eine GRUB-2-Konfigurationsdatei aus Vorlagen
`grub-mkconfig`(8) 28

`grub-mkpasswd-pbkdf2` Bestimmt verschlüsselte Kennwörter für GRUB 2
`grub-mkpasswd-pbkdf2`(1) 27

`update-grub` Aktualisiert die GRUB-2-Konfiguration (Debian)
`update-grub`(8) 28

`wipe` Löscht Dateien (oder ganze Festplatten) gründlich und endgültig
`wipe`(1) 23

Zusammenfassung

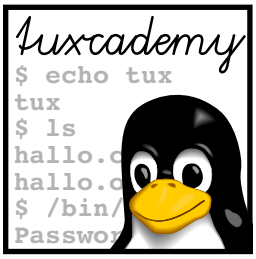
- Die physische Sicherheit von Rechnern wird gefährdet durch Einflüsse wie Speisen und Getränke, unbefugten Zugang, Feuer, Rauch, Wasser, Probleme mit der Stromversorgung, Blitzschlag, Vandalismus, Terrorismus oder Diebstahl.
- Ausrangierte Medien sollten sorgfältig gelöscht (bei Festplatten) oder vernichtet (bei optischen Datenträgern oder Magnetbändern) werden, damit Unbefugte keinen Einblick in vertrauliche Daten erhalten können.
- Minimalsysteme umgehen Probleme in nicht benötigter Software, indem die betreffende Software gar nicht erst installiert ist.
- Die gängigen Distributionen eignen sich nur eingeschränkt für Minimalsysteme.
- Geeignete BIOS-Einstellungen machen Angreifern das Leben schwerer.
- Die Bootlader LILO und GRUB erlauben Zugangsbeschränkungen für Einstellungen und Boot-Konfigurationen mit Hilfe von Kennwörtern.

Literaturverzeichnis

Cor05 Jonathan Corbet. »Who needs /dev/kmem?«, August 2005.
http://lwn.net/Articles/147901/

GSS03 Simson Garfinkel, Gene Spafford, Alan Schwartz. *Practical Unix & Internet Security*. Sebastopol, CA: O'Reilly & Associates, 2003, 3. Auflage.
http://www.oreilly.com/catalog/puis3/

Gut96 Peter Gutmann. »Secure Deletion of Data from Magnetic and Solid-State Memory«. *Proc. Sixth USENIX Security Symposium*. USENIX, 1996 S. 77–90.
http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html



3

Die Secure Shell (für Fortgeschrittene)

Inhalt

Lernziele

- Die Secure Shell (SSH) anwenden und konfigurieren können
- Netzwerkverbindungen über die SSH leiten können
- Fortgeschrittene Authentisierungsverfahren der SSH verstehen und einsetzen können

Vorkenntnisse

- Grundkenntnisse über Konfiguration und Einsatz der SSH auf LPIC-1-Niveau (siehe z. B. die Linup-Front-Schulungsunterlage *Linux-Administration II*)

3.1 Einführung

SSH (*Secure Shell*, [RFC4253]) ist ein Netzwerkprotokoll der TCP/IP-Familie. Es ermöglicht die Datenübertragung im Netz mit sicherer Authentisierung und Verschlüsselung. Zu seinen Anwendungen gehören interaktive Anmeldevorgänge, Übertragung von Dateien und die gesicherte Weiterleitung anderer Protokolle (engl. *tunneling*). SSH verwendet asymmetrische Kryptoverfahren wie RSA zur Authentisierung und zum Schlüsselaustausch und symmetrische Kryptoverfahren für die eigentliche Datenübertragung.

OpenSSH

OpenSSH, das mit den meisten Linux-Distributionen ausgeliefert wird, stellt eine frei verfügbare Implementierung dieses Protokolls dar. OpenSSH enthält einige SSH-Clients (`ssh`, `scp`, `sftp`), einen SSH-Server (`sshd`) sowie diverse Hilfsprogramme für Aufgaben wie die Verwaltung von Schlüsseln.

Einsatzmöglichkeiten

SSH ersetzt die unsicheren Protokolle TELNET, RLOGIN und RSH für interaktive Anmeldevorgänge. Zusätzlich bietet es die Möglichkeit, Dateien von einem entfernten Rechner zu kopieren und ist so ein sicherer Ersatz für RCP und viele Anwendungen von FTP.

Protokollversionen

Das SSH-Protokoll existiert in zwei Versionen, 1 und 2. Viele Clients unterstützen beide Versionen und die meisten Server können Verbindungen mit beiden Versionen entgegennehmen. Machen Sie trotzdem einen Bogen um die Version 1, die diverse Sicherheitslücken aufweist.

Auf dem Server – also dem Rechner, wo Sie sich anmelden wollen – muss der SSH-Server (`sshd`) laufen. Um mit diesem Server Verbindung aufzunehmen, brauchen Sie auf dem Client-Rechner das Programm `ssh`.

Was natürlich nicht heißt, dass Sie nicht auch irgendeinen anderen SSH-Client oder -Server benutzen könnten – es gibt diverse Implementierungen des Protokolls für die verschiedensten Plattformen. Wir reden im Rest dieses Kapitels aber nur noch über die OpenSSH.

Die Konfigurationsdateien für den `sshd` finden sich in der Regel in `/etc/ssh`, allen voran die Datei `sshd_config`. Konfigurationseinstellungen für den Client, `ssh`, finden sich systemweit in `/etc/ssh/ssh_config` und individuell für jeden Benutzer in `~/.ssh/config`. Einstellungen in der benutzerspezifischen Datei haben Vorrang.

3.2 Grundlegende Funktionalität

Hier ist ein Überblick über die grundlegende Funktionalität der OpenSSH, basierend auf der Linup-Front-Schulungsunterlage *Linux-Administration II*:

Anmelden auf entfernten Rechnern Um sich über SSH auf einem entfernten Rechner anzumelden, müssen Sie das Kommando `ssh` aufrufen, etwa so:

```
$ ssh blue.example.com                    Mit dem lokalen Benutzernamen
$ ssh hschulz@blue.example.com            Als hschulz
```

Beim ersten Verbindungsaufbau mit einem neuen entfernten Rechner müssen Sie den öffentlichen Schlüssel dieses Rechners akzeptieren. Vergewissern Sie sich bei dessen Administrator, dass der Schlüssel, den Sie angezeigt bekommen, authentisch ist.

Wenn Sie der Administrator des entfernten Rechners sind und jemand die Authentizität Ihres öffentlichen RSA-Schlüssels überprüfen möchte, dann verwenden Sie etwas wie

```
# ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key
```

um dessen »Fingerabdruck« (engl. *fingerprint*) zum Vergleich anzuzeigen. (Für die anderen Verschlüsselungsverfahren müssen Sie den passenden Dateinamen verwenden.)

Fingerabdruck

Die SSH merkt sich die öffentlichen Schlüssel von entfernten Rechnern in der Datei `~/.ssh/known_hosts`.

`~/.ssh/known_hosts`

Wenn Sie eine Verbindung zu einem Rechner aufbauen, auf dem Sie sich schon früher angemeldet haben, und eine Warnung erscheint, dann kann es sein, dass ein Unbefugter Ihnen einen falschen öffentlichen Schlüssel geschickt hat. Vielleicht hat aber auch nur der Administrator des Rechners aus irgendwelchen Gründen einen neuen öffentlichen Schlüssel generiert. Gehen Sie der Sache auf den Grund, bevor Sie die Verbindung zulassen.

Warnung

Genaugenommen entscheidet die Direktive `StrictHostKeyChecking` in der Konfiguration von `ssh` darüber, wie mit neuen oder veränderten öffentlichen Schlüsseln umgegangen werden soll:

`StrictHostKeyChecking`

`StrictHostKeyChecking ask`	*Nachfragen (Voreinstellung)*
`StrictHostKeyChecking no`	*Alle Schlüssel akzeptieren*
`StrictHostKeyChecking yes`	*Keine neuen Schlüssel akzeptieren*

Mit »`StrictHostKeyChecking yes`« können Sie nur Verbindungen zu Rechnern aufbauen, die schon in Ihrer `known_hosts`-Datei stehen. Alle anderen werden abgewiesen.

Nachdem Sie über `ssh` eine Verbindung aufgebaut haben, können Sie an dem entfernten Rechner so arbeiten, als säßen Sie davor. Sie können die Verbindung mit `exit` oder Strg+d beenden.

Wenn Sie nichts Anderes sagen, gilt bei interaktiven `ssh`-Sitzungen eine Tilde (»~«), wenn ihr in der Eingabe unmittelbar ein Zeilentrenner vorausgeht, als Sonderzeichen, mit dem Sie die `ssh` steuern können. Insbesondere bricht die Kombination »~.« die Verbindung ab, was nützlich sein kann, wenn sich am »anderen Ende« ein Programm aufgehängt hat.

Steuerung

Sie können auf dem entfernten Rechner statt interaktiven Sitzungen auch einzelne Kommandos ausführen:

einzelne Kommandos

```
$ ssh blue.example.com ls -l /home/hugo
```

(Achten Sie dabei auf eventuell unerwünschte Aktivitäten der lokalen Shell beim Auswerten des Kommandos.)

Andere nützliche Anwendungen Mit `scp` können Sie über eine SSH-Verbindung Dateien zwischen zwei Rechnern kopieren. Die Syntax ist angelehnt an `cp`:

`scp`

```
$ scp blue.example.com:hello.c .
$ scp -r subdir blue.example.com:/tmp                    Verzeichnis
$ scp hugo@blue.example.com:hello.c pink.example.com:hnew.c
```

Das Kommando `sftp` ist locker an gängige FTP-Clients angelehnt, aber verwendet eine SSH-Verbindung. Mit FTP hat es ansonsten überhaupt nichts zu tun – insbesondere können Sie es nicht verwenden, um mit einem FTP-Server zu kommunizieren.

`sftp`

Client-Authentisierung über Schlüsselpaare Statt bei der Anmeldung Ihr auf dem entfernten Rechner gültiges Kennwort einzugeben, können Sie mit `ssh-keygen` ein Schlüsselpaar erzeugen (etwa für RSA) und den öffentlichen Schlüssel auf dem entfernten Rechner hinterlegen:

Schlüsselpaar erzeugen

```
$ ssh-keygen -t rsa                                          oder dsa
```

Dabei müssen Sie eine »Passphrase« angeben, mit der der private Schlüssel verschlüsselt wird. Anschließend können Sie mit etwas wie

Passphrase

```
$ ssh-copy-id hugo@blue.example.com
```

den öffentlichen Schlüssel auf den entfernten Rechner kopieren. (Notfalls geht das auch mit `scp`.)

Vorteil

Der Vorteil bei der Verwendung eines Schlüsselpaars ist weniger der geringere Tippaufwand (statt des entfernten Kennworts müssen Sie jetzt die lokale »Passphrase« eingeben – aber siehe unten), sondern eher der Umstand, dass die Sicherheit Ihres Zugangs nicht mehr davon abhängt, dass die Administratoren des entfernten Rechners ihr Handwerk verstehen. Auf dem entfernten Rechner liegt ja nichts Geheimes mehr von Ihnen.

Mit »`PasswordAuthentication no`« und »`PubkeyAuthentication yes`« in `/etc/ssh/sshd_config` auf dem Server ist eine Anmeldung *nur noch* über Schlüsselpaare möglich. Damit sperrt man Cracker aus, die das Internet automatisch nach SSH-Servern mit schwachen Kennwörtern abklopfen.

Der `ssh-agent` Das wiederholte Eintippen der Passphrase für Ihren privaten Schlüssel können Sie vermeiden, indem Sie den `ssh-agent` verwenden. Diesem teilen Sie die Passphrase einmal mit dem Kommando `ssh-add` mit, und er merkt sie sich und stellt sie bei Bedarf dem `ssh`-Programm zur Verfügung.

ssh-add

`ssh-agent` starten

Sie können den `ssh-agent` entweder mit einem Kommando wie

```
$ ssh-agent bash
```

starten (in der neuen Shell steht er dann zur Verfügung) oder arrangieren, dass der `ssh-agent` schon beim Anmelden für Sie gestartet wird. Viele Linux-Distributionen machen das schon »ab Werk«. Mit »`ssh-add -D`« können Sie den Agenten dazu bringen, Ihre Passphrase wieder zu »vergessen«.

Der Gewinn an Bequemlichkeit ist natürlich mit einem Verlust an Sicherheit verloren. Passen Sie zum Beispiel auf, dass Ihr Notebook nicht verlorengeht, während der `ssh-agent` Ihre Passphrase kennt.

X11-Weiterleitung Wenn Sie sich auf einem entfernten Rechner anmelden und dabei die `ssh`-Option `-X` angeben, können Sie auf dem entfernten Rechner X11-Clients starten, die den lokalen X11-Server für ihre Ein- und Ausgabe verwenden.

Der Vorteil dieses Ansatzes ist, dass das X11-Protokoll so zwischen den Rechnern verschlüsselt übertragen wird. Beim »normalen« X11-Fernzugriff über die `DISPLAY`-Variable fließen die Daten dagegen im Klartext. Außerdem nutzt die SSH die üblichen Authentisierungsmechanismen der Secure Shell, während die Standardmethode nur die lasche Authentisierung des lokalen X-Servers verwendet.

Damit das Ganze funktioniert, muss beim entfernten Rechner in der `sshd`-Konfiguration die Direktive `X11Forwarding` auf `yes` gesetzt sein.

Die Parametereinstellung »`ForwardX11 yes`« in der Konfiguration des SSH-*Clients* aktiviert die X11-Weiterleitung auf dauerhafter Basis, so dass Sie die Option `-X` nicht immer angeben müssen.

Portweiterleitung SSH kann nicht nur das X11-Protokoll, sondern fast beliebige TCP-basierte Verbindungen weiterleiten und durch die verschlüsselte Verbindung »tunneln«. Mit

TCP-basierte Verbindungen weiterleiten

```
$ ssh -L 10110:mail.example.com:110 hugo@blue.example.com
```

werden Verbindungen an den lokalen TCP-Port 10110 zuerst über eine (verschlüsselte) SSH-Verbindung an den Rechner `blue.example.com` geschickt. Von da geht es (unverschlüsselt) weiter auf den TCP-Port 110 (POP3) von `mail.example.com`. Der Nutzen dieses Szenarios ist etwa wie folgt: Stellen Sie sich vor, Ihr Firewall sperrt POP3, aber läßt SSH durch. Über die Portweiterleitung kommen Sie via SSH ins interne Netz und können vom Rechner `blue.example.com` aus rein im internen Netz mit dem Mailserver reden. In Ihrem Mailprogramm geben Sie dann `localhost` und den lokalen TCP-Port 10110 als »POP3-Server« an.

Theoretisch könnten Sie auch direkt den lokalen TCP-Port 110 weiterleiten, aber dazu müssen Sie `root` sein.

Der Name des Rechners für die Weiterleitung (hier `mail.example.com`) wird aus der Sicht des SSH-Servers (hier `blue.example.com`) aufgelöst. Das heißt, eine Weiterleitung der Form

```
$ ssh -L 10110:localhost:110 hugo@blue.example.com
```

verbindet Sie mit dem Port 110 auf `blue.example.com`, nicht etwa auf Ihrem Rechner.

Wenn Sie `ssh` wie gezeigt aufrufen, bekommen Sie außer der Portweiterleitung auch eine interaktive Sitzung. Wenn Sie das nicht möchten, können Sie die Option `-N` angeben, die `ssh` auf die Weiterleitung beschränkt und keine interaktive Sitzung aufbaut.

Die Portweiterleitung funktioniert auch umgekehrt: Mit

```
$ ssh -R 10631:localhost:631 hugo@blue.example.com
```

wird der TCP-Port 10631 *auf dem SSH-Server* geöffnet und Verbindungen, die Programme dort mit diesem Port aufnehmen, über die SSH-Verbindung auf Ihren lokalen Rechner geleitet. Ihr lokaler Rechner übernimmt dann die unverschlüsselte Weiterleitung an das angegebene Ziel, hier den TCP-Port 631 auf Ihrem lokalen Rechner selbst. (Diese Form der Weiterleitung ist ungleich weniger wichtig als die über `-L`.)

3.3 Benutzer-Beschränkungen

Auf einem SSH-Server haben Sie umfangreiche Möglichkeiten, um den Zugang auf bestimmte Benutzer zu beschränken oder bestimmte Benutzer vom Zugang auszuschließen. Im einfachsten Fall können Sie einzelne Benutzer daran hindern, sich anzumelden, indem Sie deren Benutzernamen in einer `DenyUsers`-Direktive in der Konfigurationsdatei `/etc/ssh/sshd_config` aufzählen:

`DenyUsers`

```
DenyUsers hugo susi
```

schließt zum Beispiel die Benutzer `hugo` und `susi` aus. Alle anderen Benutzer dürfen sich anmelden.

Tatsächlich müssen Sie sich nicht auf Benutzernamen beschränken, sondern dürfen shell-ähnliche Suchmuster angeben: Ein Stern (`*`) steht für beliebig viele beliebige Zeichen, ein Fragezeichen für genau ein beliebiges Zeichen. (Zeichenklassen mit eckigen Klammern gibt es hier nicht.) Zum Beispiel werden mit

```
DenyUsers studi1*
```

alle Benutzer mit Namen wie `studi123`, `studi1` oder `studi1111111` ausgesperrt.

Bei Suchmustern, die einen Klammeraffen (`@`) enthalten, wird der Teil links vom `@` als Benutzername und der Teil rechts vom `@` als Rechnername betrachtet, und die beiden werden getrennt geprüft. Beachten Sie aber, dass der Benutzername sich auf den SSH-Server bezieht und der Rechnername auf den Rechner, von dem aus der betreffende Benutzer sich anmelden möchte: Steht in der Konfiguration des SSH-Servers `server.example.com` die Zeile

```
DenyUsers hugo@client.example.com
```

dann passt sie auf den Benutzer `hugo` auf `server.example.com`, wenn er versucht, sich von `client.example.com` anzumelden – egal was sein Benutzername auf `client.example.com` sein mag.

`AllowUsers` Mit `AllowUsers` können Sie umgekehrt eine Positivliste von Benutzernamen (bzw. Suchmustern für Benutzernamen) angeben. Nur diese Benutzer dürfen sich dann anmelden:

```
AllowUsers susi fritz
```

(Rechnernamen sind hier auch erlaubt, in Analogie zu `DenyHosts`.)

`AllowGroups` `DenyGroups` `AllowGroups` und `DenyGroups` erlauben es Ihnen, pauschal die Mitglieder bestimmter Gruppen zuzulassen oder auszusperren. Dabei werden sowohl die primäre als auch die zusätzlichen Gruppen jedes Benutzers betrachtet. Suchmuster dürfen Sie auch benutzen, allerdings gibt es keine Sonderbehandlung für Klammeraffen und »Rechnernamen«.

Wenn mehrere dieser Direktiven in der Konfiguration auftauchen, werden sie in der Reihenfolge `DenyUsers`, `AllowUsers`, `DenyGroups`, `AllowGroups` ausgewertet. Der erste Treffer zählt; bei etwas wie

```
DenyUsers *u*
AllowUsers susi
```

wird `susi` also abgewiesen, da die `AllowUsers`-Direktive nicht mehr zum Tragen kommt.

Wenn Sie meinen, dass eine Kombination dieser Direktiven keinen großen Sinn ergibt, dann haben Sie vermutlich Recht. Es ist besser, die Dinge einfach zu halten.

`PermitRootLogin` Die Direktive `PermitRootLogin` bestimmt, ob der Benutzer `root` sich direkt über SSH anmelden kann (wenn nein, dann müssen Sie sich gegebenenfalls als normaler Benutzer anmelden und mit `su` oder `sudo` zu `root` werden). »`PermitRootLogin no`« ist dasselbe wie »`DenyUsers root`«, und der Standardfall ist »`PermitRootLogin yes`«.

Es gibt noch ein paar andere interessante Werte für `PermitRootLogin`: »`without-password`« erlaubt `root` nicht etwa die Anmeldung ganz ohne Kennwort, sondern verbietet `root` eine kennwortbasierte Anmeldung und besteht statt dessen auf einer Anmeldung per Schlüsselpaar – alle anderen Benutzer dürfen sich aber noch per Kennwort anmelden. Mit »`forced-commands-only`« bekommt `root` keine interaktive Shell, sondern muss sich über ein Schlüsselpaar anmelden, wobei im öffentlichen Schlüssel ein festes Kommando angegeben ist, das auf jeden Fall ausgeführt wird. (Über feste Kommandos sagen wir in Abschnitt 3.4.4 noch mehr.)

Zeitweise Sperrung

Vielleicht möchten Sie einen Rechner zeitweise für SSH-Logins von »gewöhnlichen« Benutzern sperren – etwa weil Sie Systemarbeiten ausführen. Am einfachsten geht das, indem Sie eine Datei namens `/etc/nologin` anlegen. Wenn diese Datei existiert, darf sich nur noch `root` anmelden (ein geeignetes `PermitRootLogin` mal vorausgesetzt); alle anderen Benutzer werden abgewiesen, und der Inhalt der Datei wird als Erklärung angezeigt: Mit

```
# cat /etc/nologin
Wartungsarbeiten bis 12.15 Uhr, sorry.
```

sieht ein SSH-Client etwas wie

```
$ ssh hugo@blue.example.com
hugo@blue.example.com's password: hugo123
Wartungsarbeiten bis 12.15 Uhr, sorry.

Connection closed by blue.example.com
```

Wenn `/etc/nologin` Ihnen noch von `shutdown` und/oder PAM bekannt ist: Herzlichen Glückwunsch!

Übungen

3.1 [!2] Überzeugen Sie sich, dass die Direktiven `AllowUsers` und `DenyUsers` so funktionieren wie beschrieben.

3.2 [1] Warum bewirkt die Direktive »`PermitRootLogin yes`« nicht dasselbe wie »`AllowUsers root`«?

3.3 [2] Probieren Sie aus, ob »`PermitRootLogin no`« so funktioniert wie angegeben. Wie sieht es aus mit »`PermitRootLogin without-password`«?

3.4 [2] Warum sollte man »`PermitRootLogin no`« verwenden? Welche Gefahr geht davon aus? Was sind mögliche Alternativen? Diskutieren Sie.

3.5 [1] Wie würden Sie gewöhnliche Benutzer für eine Stunde »aussperren«, ohne dass Sie sich daran erinnern müssen, den Zugang danach wieder freizugeben?

3.4 Tipps und Tricks

3.4.1 Benutzer-Konfiguration für verschiedene Server

Im täglichen Leben versucht das OpenSSH-Clientprogramm `ssh`, so weit es kann, das Richtige zu tun. Zum Beispiel nimmt es an, dass Ihr Benutzerkonto auf einem entfernten Server genauso heißt wie Ihr lokales Konto – jedenfalls solange Sie nicht ausdrücklich etwas Anderes sagen. Allerdings kann es umständlich sein, immer wieder Kommandos wie

```
$ ssh hschulz@server23.example.net
```

eintippen zu müssen, wenn ein einfaches

```
$ ssh s23
```

eigentlich auch reichen würde. Zum Glück ist das kein Problem!

Sie können in Ihrer `ssh`-Konfigurationsdatei `~/.ssh/config` nämlich Abschnitte der Form

```
Host s23
  HostName server23.example.net
  User hschulz
```

Host haben, um Konfigurationseinstellungen vorzunehmen, die nur für Verbindungen zu s23 gelten. Allgemein gesagt führt die Host-Direktive dazu, dass alle folgenden Einstellungen nur für Verbindungen zu der genannten Gegenstelle gelten – bis zur nächsten Host-Direktive oder dem Dateiende.

Hinter »Host« dürfen mehrere Namen stehen. Auch Suchmuster mit »*« und »?« sind erlaubt. Mit »Host *« können Sie Vorgaben machen, die für alle Verbindungen gelten.

Vorgehensweise Das Programm ssh bestimmt die Werte von Konfigurationseinstellungen, indem es die Konfigurationsdatei vom Anfang her betrachtet. Das erste Auftreten einer Direktive – gegebenenfalls in einem passenden Host-Block – zählt. Sie sollten also aufpassen: Eine Konstruktion wie

```
Compression yes                                   Daten komprimieren …

Host server.example.com
  Compression no                          … außer für diesen Rechner!?
```

funktioniert also nicht wie beabsichtigt, da das »Compression yes« zuerst gefunden wird und die Suche danach abbricht. Auch Verbindungen zu server.example.com verwenden also Datenkomprimierung. Richtig wäre

```
Host server.example.com
  Compression no

Host *                                                Alle außer server
  Compression yes
```

3.4.2 Feinheiten des Protokolls

Wie eingangs erwähnt gibt es zwei Hauptversionen des SSH-Protokolls, einfallsreich als »1« und »2« bezeichnet. Die Protokollversion 1 interessiert uns nicht weiter, außer dass wir Sie nachdrücklich davor warnen wollen, sie zu benutzen.

Die Version 1 des SSH-Protokolls hat einige dicke Macken, etwa die, dass ein Angreifer unter bestimmten Umständen seine eigenen Daten in den verschlüsselten Datenstrom einschleusen kann (CVE-1999-1085). Eine genauere Beschreibung dieses Angriffs findet sich unter http://www.kb.cert.org/vuls/id/13877.

Falls Sie sich detaillierter über die Unterschiede zwischen SSH 1 und SSH 2 informieren möchten, konsultieren Sie http://www.snailbook.com/faq/ssh-1-vs-2.auto.html.

SSH-Server Sie können Ihren SSH-Server daran hindern, SSH-1-Verbindungen zu akzeptieren, indem Sie in die sshd_config-Datei die Zeile

```
Protocol 2
```

einfügen. Damit redet er nur mit Clients, die diese Version verstehen können.

Clientseite Auf der Clientseite können Sie mit

```
Protocol 2
```

weit vorne in der Konfiguration erreichen, dass Ihr Client nur diese Version des Protokolls zu sprechen bereit ist. (Denken Sie daran: Der erste Treffer für eine Konfigurationseinstellung zählt.)

Sie können als Systemadministrator nicht erzwingen, dass alle Ihre Benutzer niemals die Protokoll-Version 1 verwenden, da `ssh` die benutzerspezifische Konfiguration in `.ssh/config` vor der systemweiten Konfiguration in `/etc/ssh/ssh_config` anschaut – und in jedem Fall ein Parameter wie »`-1`« auf der Kommandozeile Vorrang vor allem anderen hat. Kategorische Vorgaben funktionieren nur auf dem Server: Für Ihre eigenen Server können Sie SSH-1-Unterstützung (wie gezeigt) global deaktivieren, aber für andere natürlich nicht.

Prinzipiell könnten Sie (wenn Sie entweder sehr dumm oder tollkühn wären), mit

```
Protocol 2,1                                    1,2 ginge auch, hu brr
```

angeben, dass Ihr Client zuerst versuchen soll, über SSH 2 Kontakt mit dem Server aufzunehmen, um dann – falls das nicht klappt –, auf die Version 1 zurückzufallen. Auf der Serverseite führt diese Einstellung dazu, dass der Server bereit ist, je nach Wunsch des Clients beide Protokollversionen zu sprechen (hier gibt es keine Priorisierung – der Kunde ist König).

Sollten Sie in die missliche Lage kommen, in Ihrem Client etwas wie »`Protocol 2,1`« benutzen zu müssen, dann grenzen Sie dies per `Host` gezielt auf die betreffenden Server ein. Versuchen Sie auch, die Betreiber der betreffenden Server von einem Upgrade zu überzeugen. (Sie müssen ja nicht gleich die Herren vom Russen-Inkasso vorbeischicken, aber ernsthaft, Leute: SSH 2 gibt es seit dem letzten Jahrtausend.)

3.4.3 Netz und doppelter Boden

Ein wichtiges Einsatzgebiet von SSH ist natürlich die Wartung entfernter Rechner – wobei »entfernt« alles bedeuten kann von »im Serverschrank draußen auf dem Gang« bis »am anderen Ende der Welt«. Je nachdem, wo der entfernte Rechner steht und wie einfach es ist, notfalls an eine direkt angeschlossene Konsole zu kommen, ist es nötig, mehr oder weniger vorsichtig zu sein – vor allem, wenn Sie an der Konfiguration des `sshd` selbst herumbasteln. Schließlich ist es leicht möglich, sich auszusperren, und wenn das eine dreistündige Autofahrt oder eine ebensolange Telefonsitzung mit den tranigen Operatoren vor Ort bedeutet, bis der Zugang wieder hergestellt ist, dann ist es sicherlich besser, es gar nicht erst so weit kommen zu lassen.

Wenn Sie an der `sshd`-Konfiguration auf einem entfernten Rechner Änderungen machen wollen, sollten Sie als Erstes in einem anderen Terminal-(unterfenster) eine zweite Sitzung auf den betreffenden Rechner öffnen. Es zeigt sich nämlich, dass diese Sitzung Konfigurationsneuladevorgänge und sogar `sshd`-Neustarts souverän überlebt. Sollten Sie also im einen Fenster den Ast absägen, auf dem Sie bisher gesessen haben, und sich nicht mehr anmelden können, dann steht Ihnen die andere Sitzung zur Verfügung, um alles wieder in Ordnung zu bringen.

Übungen

3.6 [2] Vergewissern Sie sich, dass das eben Gesagte tatsächlich stimmt: Melden Sie sich zweimal über SSH auf einem »entfernten« Rechner an, etwa als `susi@blue.example.com`. Verwenden Sie eine der Sitzungen, um `root`-Rechte anzunehmen, in `/etc/ssh/sshd_config` die Zeile

```
DenyUsers susi
```

einzutragen und den SSH-Server dann mit

```
# service ssh reload                                    oder so ähnlich
```

anzustupsen, damit er die geänderte Konfiguration einliest. Überzeugen Sie sich, dass (a) Sie sich in dieser Sitzung zwar ab-, aber nicht wieder anmelden können, aber (b) die andere Sitzung nach wie vor funktioniert.

3.7 [2] Was passiert mit der zweiten Sitzung, wenn Sie in der einen Sitzung den SSH-Server komplett anhalten (mit etwas wie »`service ssh stop`)? Was passiert, wenn Sie den SSH-Server neu starten?

3.4.4 Spaß mit öffentlichen Schlüsseln

Bisher hatten wir gesehen, dass die Authentisierung über Schlüsselpaare in erster Linie der Verbesserung der Sicherheit dient: Auf dem entfernten Server muss nichts Geheimes mehr gespeichert werden, so dass Sie sich nicht davon abhängig machen, dass den dortigen Administratoren kein Missgeschick passiert, bei dem Ihr (verschlüsseltes) Kennwort einem Angreifer in die Hände fällt. Statt dessen müssen Sie Ihren privaten Schlüssel niemals aus der Hand geben, was natürlich ein gravierender Vorteil ist.

Wenn Sie ganz besonders auf Sicherheit bedacht sind, können Sie Ihren privaten Schlüssel zum Beispiel auf einer Chipkarte speichern statt auf Ihrem Computer. Dort ist er dann sicher vor allfälligen Viren, Würmern oder Trojanern – nicht dass die unter Linux bisher ein großes Problem wären, aber man weiß ja nie …

Um optimale Sicherheit zu gewährleisten, werden private SSH-Schlüssel außerdem noch mit einer »Passphrase« verschlüsselt, die es Unbefugten schwerer machen soll, mit einem privaten Schlüssel, der ihnen irgendwie in die Hände fällt, tatsächlich etwas anzufangen.

Diese Verschlüsselung ist aber manchmal eher ein Hindernis. Stellen Sie sich zum Beispiel vor, Sie möchten irgendwann in der Nacht eine Sicherheitskopie von Ihrem Datenbankserver machen, indem Sie einen Abzug des Datenbankinhalts auf einen entfernten Rechner kopieren. Mit SSH ist das im Grunde sehr einfach: Sie starten vom entfernten Rechner aus per `cron` und SSH auf dem Datenbankserver ein Programm, das die Datenbank liest und das Ergebnis auf seine Standardausgabe schreibt. Die Standardausgabe wird mit SSH über den verschlüsselten Kanal auf den entfernten Rechner geleitet und dort geeignet archiviert. Der einzige Haken besteht darin, dass Sie auf dem entfernten Rechner, der die Sicherheitskopie anstößt, theoretisch die Passphrase für Ihren privaten Schlüssel eingeben müßten, damit er mit dem Datenbankserver reden kann – und `cron`-Jobs haben bekanntlich keine interaktive Eingabe.

Schlüssel ohne Passphrase

Die naheliegende Abhilfe ist, einen privaten Schlüssel ohne Passphrase zu benutzen. Das funktioniert ohne Weiteres – drücken Sie einfach ↩, wenn `ssh-keygen` Sie nach der Passphrase für den neuen privaten Schlüssel fragt –, aber birgt natürlich das Problem, dass der private Schlüssel einem Cracker beliebigen Zugriff auf den Datenbankserver einräumt. Nicht so gut.

Sie können die Passphrase auf einem privaten Schlüssel auch nachträglich entfernen. Rufen Sie dazu `ssh-keygen` mit der Option `-p` auf und drücken Sie ↩, wenn Sie nach der neuen Passphrase gefragt werden:

```
$ ssh-keygen -p
Enter file in which the key is (/home/hugo/.ssh/id_rsa): ↩
Enter old passphrase: geheim
Key has comment '/home/hugo/.ssh/id_rsa'
Enter new passphrase (empty for no passphrase): ↩
```

```
Enter same passphrase again: ↩
Your identification has been saved with the new passphrase.
$ _
```

Festes Kommando

Um dieses Problem abzumildern, können Sie dafür sorgen, dass der öffentliche Schlüssel auf dem Server nur dafür zu gebrauchen ist, das Kommando zum Starten der Sicherheitskopie aufzurufen. (Dass ein Cracker das tun kann, ist zwar möglicherweise schlimm genug, aber noch nicht so schlimm wie beliebige Kommandos ausführen dürfen.) Dazu müssen Sie in der Datei `.ssh/authorized_keys` im betreffenden Benutzerkonto auf dem Server den Schlüssel finden und eine entsprechende Klausel an den Anfang setzen:

```
$ ssh hugo@blue.example.com
blue$ cd .ssh
blue$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAA◁◁◁◁◁                    Eine lange Zeile
blue$ vi authorized_keys
command="date" ssh-rsa AAAAB3NzaC◁◁◁◁◁
blue$ exit                                              SSH-Sitzung beenden
$ ssh hugo@blue.example.com ls -l
Mo 9. Dez 23:43:36 CET 2013 »date« wird ausgeführt
```

Wenn ein öffentlicher Schlüssel auf dem Server mit `command="…"` ein Kommando definiert, dann wird keine interaktive Shell gestartet, sondern lediglich das angegebene Kommando ausgeführt, egal was der Client gerne machen würde.

Wenn der Client ein Kommando überträgt, wird das dem tatsächlich aufgerufenen (festen) Kommando in der Umgebungsvariable `SSH_ORIGINAL_COMMAND` zugänglich gemacht. Dieses kann dann entscheiden, ob und wie es mit dieser Information umgeht.

Auch wenn er auf diese Weise keine beliebigen Kommandos ausführen kann, könnte ein Angreifer trotzdem X11- oder Portweiterleitung aktivieren und auf diese Weise unter Umständen Zugriff auf Rechner oder Ressourcen bekommen, die vom SSH-Serverrechner aus zugänglich sind, vom Client aus aber nicht. Um das auszuschließen, muss der öffentliche Schlüssel außer der `command`-Klausel noch die Direktiven `no-port-forwarding` und `no-x11-forwarding` enthalten. (Wenn Sie mehrere Direktiven angeben, müssen Sie sie durch Kommas trennen.)

Mit `from` können Sie eine Liste von Rechnernamen und/oder IP-Adressen angeben, die für Verbindungen mit diesem öffentlichen Schlüssel als Clients in Frage kommen: Ein Eintrag in `authorized\_keys` wie

```
command="date",from="192.168.1.*" ssh-rsa AAAAB3NzaC◁◁◁◁◁
```

würde mit diesem Schlüssel authentisierten Clients nur die Ausführung des Kommandos `date` erlauben, und zwar nur dann, wenn ihre IP-Adresse im Subnetz `192.168.1.0/24` liegt.

Standardmäßig werden Kommandos in einem Pseudo-Terminal (PTY) ausgeführt, wenn der Client darum nachsucht. Pseudo-Terminals werten Steuerzeichen aus und bemühen sich auch anderweitig, den darin laufenden Programmen vorzumachen, ihre Standard-Ein- und -Ausgabe wäre mit einem »echten« Terminal verbunden[1]. Das heißt aber, dass Zeichen möglicherweise verschluckt werden, wenn das PTY sie für Steuerzeichen hält. Wenn Sie sichergehen wollen, dass ein Schlüssel nur für direkte Sitzungen ohne PTY benutzt werden kann, dann fügen Sie die Direktive `no-pty` hinzu.

[1] Das ist nützlich für interaktive Shellsitzungen, damit Programme wie der `vi`, die den ganzen Bildschirm übernehmen wollen, vernünftig laufen können.

Übungen

3.8 [!2] Sorgen Sie dafür, dass Sie auf einem entfernten Rechner nur ein unverfängliches Kommando (etwa `id` oder `date`) ausführen können, wenn Sie sich mit Ihrem Schlüsselpaar anmelden. Überzeugen Sie sich, dass die Einschränkung auch gilt, wenn Sie im `ssh`-Aufruf ein anderes Kommando angeben.

3.9 [3] Wie können Sie demselben öffentlichen Schlüssel mehrere verschiedene feste Kommandos erlauben? (Mit anderen Worten: Sorgen Sie dafür, dass Sie über Ihren öffentlichen Schlüssel *sowohl* `id` *als auch* `date` ausführen können – Kommandos wie

```
$ ssh blue.example.com date
$ ssh blue.example.com id
```

sollen erlaubt sein, aber

```
$ ssh blue.example.com rm -rf .
```

nicht. Probieren Sie Ihre Lösung aus, am besten mit einem etwas weniger gefährlichen »verbotenem« Kommando.)

3.5 OpenSSH-Zertifikate

3.5.1 Überblick

Die Authentisierung über Schlüsselpaare ist flexibel und (vergleichsweise) sicher, aber auch nervend zu konfigurieren: Sie müssen Ihren öffentlichen Schlüssel auf jeden einzelnen Rechner kopieren, auf dem Sie sich anmelden wollen. Umgekehrt müssen Sie auch den öffentlichen Rechner-Schlüssel jedes dieser Rechner einzeln annehmen (und prüfen!!!). Das ist nicht wirklich bequem. Es wäre viel schöner, wenn die OpenSSH es erlauben würde, einerseits die Authentizität von öffentlichen Rechner-Schlüsseln a priori zu bestätigen und andererseits das Anmelden auf entfernten Rechnern auch ohne das vorherige Kopieren öffentlicher Benutzer-Schlüssel zu gestatten.

Zertifikate

Genau dafür sind **Zertifikate** gedacht. Sie können nämlich spezielle Schlüsselpaare generieren, die als »Zertifizierungsstelle« fungieren. Damit müssen Sie auf einem Rechner nur noch den öffentlichen Schlüssel eines solchen Schlüsselpaars installieren, damit alle Benutzer mit einem mit dem dazugehörigen privaten Schlüssel »signierten« Zertifikat (auf der Basis eines gewöhnlichen Benutzer-Schlüsselpaars) sich dort anmelden können, ohne vorher ihren öffentlichen Schlüssel hinterlegt zu haben. Umgekehrt kann ein Rechner, der über den öffentlichen Schlüssel einer Zertifizierungsstelle verfügt, ohne Rückfrage Verbindungen zu allen Rechnern aufbauen, deren öffentliche Rechner-Schlüssel von der Zertifizierungsstelle beglaubigt wurden.

Die »Zertifikate« der OpenSSH haben nichts mit den ansonsten verbreiteten X.509-Zertifikaten zu tun, wie sie zum Beispiel von OpenSSL und OpenVPN benutzt werden.

Eigentlich handelt es sich dabei nur um eine Spezialanwendung der ohnehin vorhandenen SSH-Schlüsselpaare. Deswegen müssen Sie einige Einschränkungen in Kauf nehmen: Zum Beispiel gibt es keine Hierarchie von Zertifierungsstellen, Unter-Zertifizierungsstellen und so weiter wie bei X.509 – bei der OpenSSH können beglaubigte Schlüsselpaare nicht verwendet werden, um weitere Schlüsselpaare zu beglaubigen. Auf der anderen Seite müssen Sie sich auch nicht mit den komplizierten Konfigurationsdateien und Kommandos von OpenSSL herumschlagen.

Um OpenSSH-Zertifikate verwenden zu können, müssen Sie auf dem Client und auf dem Server mindestens die OpenSSH-Version 5.4 einsetzen. Das ist möglicherweise für ältere Linux-Distributionen der Enterprise-Klasse ein Problem. In OpenSSH 6.0 hat das Format von Zertifikaten sich geändert, und aktuelle Zertifikate können mit älteren OpenSSH-Versionen nicht mehr benutzt werden.

Für die LPI-Prüfung 202 sind OpenSSH-Zertifikate nicht von Belang. Falls Sie nur die Prüfung machen wollen und ein kleiner Blick über den Tellerrand Sie nicht interessiert, können Sie den Rest dieses Abschnitts überspringen.

3.5.2 Benutzer-Schlüssel beglaubigen

Betrachten wir zuerst die eine Richtung des Vorgangs: Wir möchten vermeiden, öffentliche Schlüssel von Benutzern auf alle Rechner kopieren zu müssen, wo diese Benutzer sich anmelden können sollen. Statt dessen generieren wir ein Schlüsselpaar für eine »Zertifizierungsstelle«, mit der wir dann »Zertifikate« für Benutzer ausstellen können. Auf der Basis dieser Zertifikate können Benutzer sich dann auf entfernten Rechnern anmelden, ohne zuvor dort ihren öffentlichen Schlüssel hinterlegt zu haben – solange auf dem betreffenden Rechner der öffentliche Schlüssel der Zertifizierungsstelle zur Verfügung steht, um die Authentizität des Zertifikats zu prüfen.

Beginnen wir mit dem Schlüsselpaar für die Zertifizierungsstelle. Dieses wird mit `ssh-keygen` erzeugt wie ein ganz gewöhnliches SSH-Schlüsselpaar:

```
$ ssh-keygen -f .ssh/ca-key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): secret
Enter same passphrase again: secret
Your identification has been saved in ca-key.
Your public key has been saved in ca-key.pub.
```

Im wirklichen Leben möchten Sie natürlich eine etwas komplexere Passphrase benutzen.

Wichtig: Auf dieses Schlüsselpaar sollten Sie besonders gut aufpassen, denn ein Angreifer, dem es in die Hände fällt, kann damit beliebige andere OpenSSH-Schlüsselpaare signieren und so unter dem Strich beliebige Benutzeridentitäten auf allen Rechnern annehmen, die Ihrer Zertifizierungsstelle vertrauen. Von Rechts wegen sollten Sie die üblichen Vorsichtsmaßnahmen für Zertifizierungsstellen beachten und Schlüssel nur auf einem Rechner signieren, der selber nicht am Netz ist und bei Nichtgebrauch sicher weggeschlossen wird. Zuallermindestens sollten Sie dieses Schlüsselpaar aber auf einen USB-Stick verschieben und diesen bei Nichtgebrauch sicher wegschließen. Eine gut verwahrte Sicherheitskopie (oder zwei oder drei) wäre bestimmt auch kein Fehler.

Anschließend können Sie mit diesem Schlüsselpaar einen beliebigen öffentlichen OpenSSH-Schlüssel signieren:

```
$ ssh-keygen -s ~/.ssh/ca-key -I "Hugo Schulz" -n hugo \
>   -V +52w -z 0 ~/.ssh/id_rsa.pub
Enter passphrase: secret
Signed user key /home/hugo/.ssh/id_rsa-cert.pub: id "Hugo Schulz"▷
 ◁ serial 0 for hugo▷
 ◁ valid from 2013-12-11T15:08:00 to 2014-12-10T15:09:49
```

(Die Passphrase hier ist die für das Schlüsselpaar der Zertifizierungsstelle.) Dieses Kommando erzeugt ein »Zertifikat« in der Datei `.ssh/id_rsa-cert.pub`, das standardmäßig 52 Wochen lang gültig ist und eine Anmeldung als `hugo` erlaubt.

Mit »`-z 0`« wird die Seriennummer 0 für das Zertifikat festgelegt. Das ist auch der Standardwert, wenn die Option `-z` überhaupt nicht angegeben wurde; wenn Sie es vermeiden wollen, ausschließlich Zertifikate mit dieser Seriennummer auszustellen (warum das Keine Gute Idee ist, sehen wir noch), müssen Sie sich selbst darum kümmern, hier den richtigen Wert einzusetzen. Das ist ziemlich unbequem und das Erstellen von Zertifikaten damit ein erstklassiger Kandidat für die Automatisierung durch ein Shellskript. (Siehe hierzu auch Übung 3.11.)

Übungen

3.10 [!2] Legen Sie wie oben beschrieben ein Schlüsselpaar an, das als »Zertifizierungsstelle« fungiert. Signieren Sie damit Ihren öffentlichen Schlüssel.

3.11 [3] Schreiben Sie ein Shellskript namens `sign-ssh-key`, das den Dateinamen eines öffentlichen SSH-Schlüssels auf der Kommandozeile übernimmt und den Benutzer interaktiv nach der Kennung für den Schlüssel (`ssh-keygen`-Option `-I`) sowie dem Prinzipal (`ssh-keygen`-Option `-n`) fragt. Die Seriennummer soll automatisch bestimmt werden, und Seriennummer, Kennung und Prinzipal sollen an eine Protokolldatei (etwa `~/.ssh/certs-issued`) angehängt werden.

3.5.3 OpenSSH-Zertifikate für Benutzer verwenden

Damit ein entfernter Rechner das Zertifikat akzeptiert, muss er über den öffentlichen Schlüssel der Zertifizierungsstelle verfügen, damit er die Signatur der Zertifizierungsstelle auf dem Zertifikat verifizieren kann. Konkret heißt das, die Datei `.ssh/ca-key.pub` muss auf den betreffenden Rechner kopiert und der OpenSSH bekannt gemacht werden. Dabei gibt es zwei Fälle:

Ein einzelnes Benutzerkonto Damit (potentiell) beliebige Benutzer mit einem Zertifikat sich mit einem bestimmten Benutzerkonto auf dem entfernten Rechner anmelden können, muss der öffentliche Schlüssel der Zertifizierungsstelle in der Datei `.ssh/authorized_keys` im dazugehörigen Heimatverzeichnis hinterlegt und als Schlüssel einer Zertifizierungsstelle gekennzeichnet werden. Das könnte zum Beispiel so gehen:

```
$ scp .ssh/ca-key.pub blue.example.com:.ssh/ca-key.pub
$ ssh hugo@blue.example.com
blue$ whoami
hugo
blue$ cd .ssh
blue$ echo "cert-authority $(cat ca-key.pub)" >>authorized_keys
```

Anschließend wird jeder von der Zertifizierungsstelle beglaubigte Schlüssel akzeptiert, dessen Zertifikat das Anmelden als `hugo` gestattet.

Natürlich können in der `authorized_keys`-Datei außerdem noch »echte« öffentliche Schlüssel stehen.

Der ganze Rechner Um Zugriff auf beliebige Benutzerkonten zu erlauben – natürlich immer noch gemäß der Angaben im Zertifikat –, muss der öffentliche Schlüssel in einer Datei stehen, deren Name durch den Parameter der Direktive `TrustedUserCAKeys` in der `sshd`-Konfiguration angegeben wird. Sie könnten zum Beispiel eine Datei `/etc/ssh/ca-keys` erzeugen:

```
blue$ /bin/su -
blue# cat ~hugo/.ssh/ca-key.pub >>/etc/ssh/ca-keys
```

Anschließend müssen Sie nur noch dafür sorgen, dass die `sshd`-Konfigurationsdatei die Zeile

```
TrustedUserCAKeys /etc/ssh/ca-keys
```

enthält.

Wie der Plural andeutet, können Sie die öffentlichen Schlüssel mehrerer Zertifizierungsstellen in diese Datei schreiben – wohlgemerkt nur die reinen öffentlichen Schlüssel *ohne* ein davorgesetztes `cert-authority`.

Grundsätzlich können Sie Zertifikate auch ohne die Angabe eines »Prinzipals«, also eines Benutzerkontos, für das sie zu gebrauchen sein sollen, ausstellen. Solche Zertifikate können Sie allerdings nicht benutzen, wenn der öffentliche Schlüssel der dazugehörigen Zertifizierungsstelle systemweit (mit `TrustedUserCAKeys`) installiert ist – ansonsten könnte der Zertifikatsinhaber potentiell die Identität jedes Benutzers auf dem Rechner annehmen. Der öffentliche Schlüssel der Zertifizierungsstelle muss also direkt im »Zielkonto« vorliegen.

Es ist möglich, in die Zertifikate diverse Optionen aufzunehmen, die denen entsprechen, die Sie sonst in der `authorized_keys`-Datei auf dem Zielrechner angeben würden. Wenn Sie zum Beispiel ein Zertifikat mit dem Kommando

```
$ ssh-keygen -s … -Oforce-command=date \
>   -Osource-address=192.168.56.101 ...
```

ausstellen, dann kann es nur dafür benutzt werden, von dem Rechner mit der IP-Adresse `192.168.56.101` aus das Kommando `date` auszuführen.

Auch die Zeilen in `authorized_keys`, die den öffentlichen Schlüssel einer Zertifizierungsstelle enthalten, dürfen weitere Restriktionen wie feste Kommandos enthalten. Bei Kollisionen haben die Angaben in Zertifikaten Vorrang.

Übungen

3.12 [!2] Installieren Sie in Ihrem Benutzerkonto auf einem entfernten Rechner in der Datei `authorized_keys` den öffentlichen Schlüssel der Zertifizierungsstelle. Entfernen Sie, falls nötig, Ihren eigenen öffentlichen Schlüssel oder kommentieren Sie ihn aus (mit einem »#« am Zeilenanfang). Überzeugen Sie sich, dass Sie sich trotzdem mit Ihrem öffentlichen Schlüssel auf dem entfernten Rechner anmelden können.

3.13 [2] (Fortsetzung der vorigen Aufgabe.) Kopieren Sie auf dem entfernten Rechner den öffentlichen Schlüssel Ihrer Zertifizierungsstelle nach `/etc/ssh/ca-keys`. Entfernen Sie ihn dann aus der Datei `authorized_keys` in Ihrem Benutzerkonto (oder kommentieren Sie ihn aus). Konfigurieren Sie den `sshd` so, dass er die Datei `/etc/ssh/ca-keys` nach Schlüsseln zur Verifikation von Zertifikaten durchsucht. Überzeugen Sie sich, dass Sie sich noch immer mit Ihrem öffentlichen Schlüssel auf dem entfernten Rechner anmelden können.

3.14 [2] Angenommen, Sie haben (etwa wie in der vorigen Aufgabe) den öffentlichen Schlüssel der Zertifizierungsstelle systemweit installiert. Prüfen Sie, was passiert, wenn Sie sich mit einem Schlüssel anmelden wollen, dessen Zertifikat keinen Prinzipal enthält (weil beim Signieren die Option `-n` nicht angegeben wurde).

3.15 [2] Die Zertifikate sind unter der Kontrolle ihrer Inhaber. Ist es nicht gefährlich, wenn sie Optionen wie feste Kommandos für das entfernte System enthalten? Könnten die Inhaber diese Optionen nicht ändern?

3.5.4 Rechner-Schlüssel und -Zertifikate

Damit ein entfernter Rechner statt eines (vom Benutzer zu prüfenden und zu bestätigenden) öffentlichen Rechner-Schlüssels ein (mit dem vorher installierten öffentlichen Schlüssel der Zertifizierungsstelle überprüfbares) Zertifikat verwendet, müssen Sie zunächst den öffentlichen Rechner-Schlüssel mit dem privaten Schlüssel Ihrer Zertifizierungsstelle signieren:

```
$ scp blue.example.com:/etc/ssh/ssh_host_ecdsa_key.pub .
$ ssh-keygen -s ~/.ssh/ca-key -I "blue.example.com" \
>   -h -n blue.example.com \                          -h und -n sind wichtig
>   -V +52w ssh_host_ecdsa_key.pub
```

Hier müssen Sie unbedingt mit der Option `-n` den Rechnernamen als »Prinzipal« angeben. Ansonsten könnte das Zertifikat jeden beliebigen Rechner in Ihrem Netz – auch einen von einem Angreifer eingeschleppten – authentisieren.

Die resultierende Datei `ssh_host_ecdsa_key-cert.pub` kopieren Sie dann auf den Rechner `blue.example.com` zurück (am besten nach `/etc/ssh`) und in dessen `sshd`-Konfigurationsdatei die Zeile

```
HostCertificate /etc/ssh/ssh_host_ecdsa_key-cert.pub
```

einbauen.

.ssh/known_hosts

Anschließend können Sie lokal alle Verweise auf den Rechner `blue.example.com` (Name und IP-Adresse) aus Ihrer Datei `.ssh/known_hosts` entfernen. Statt dessen fügen Sie den öffentlichen Schlüssel der Zertifizierungsstelle wie folgt hinzu:

```
$ cd ~/.ssh
$ echo "@cert-authority *.example.com $(cat ca-key.pub)" >>known_hosts
```

Achten Sie auf das »`*.example.com`«; es sorgt dafür, dass der Schlüssel für die Zertifikate aller Rechner in der Domain `example.com` zu gebrauchen ist (und keine anderen). Maßgeblich dabei ist der Name, den Sie eingeben, um den entfernten Rechner anzusprechen. Sie können auch mehrere Rechnernamen und -suchmuster angeben, Sie müssen sie aber durch Kommas trennen.

Wenn Sie anschließend versuchen, eine Verbindung zu `blue.example.com` aufzubauen, sollte das ohne die sonst übliche Bestätigungsrückfrage für dessen öffentlichen Schlüssel klappen.

systemweite Vorgabe

Sie können ganz entsprechend auch eine systemweite Vorgabe machen, indem Sie den öffentlichen Schlüssel der Zertifizierungsstelle in die Datei `/etc/ssh/ssh_known_hosts` eintragen. Die Syntax ist dieselbe.

Niemand zwingt Sie übrigens dazu, die Zertifikate für Benutzer und die für Rechner mit demselben Schlüsselpaar als Zertifizierungsstelle zu beglaubigen – solange die richtigen öffentlichen Schlüssel am richtigen Platz installiert sind, können sie durchaus verschieden sein.

Übungen

3.16 [!2] Konfigurieren Sie Ihr System (den lokalen und den entfernten Rechner) so, dass öffentliche Rechner-Schlüssel über Zertifikate akzeptiert werden, Benutzer also nicht mehr gebeten werden, Schlüssel zu prüfen und anzuerkennen. (Für Sonderpunkte: Sorgen Sie dafür, dass das für den Rechnernamen *und* die IP-Adresse des entfernten Rechners funktioniert.)

Kommandos in diesem Kapitel

scp Sicheres Dateikopierprogramm auf SSH-Basis scp(1) 35
sftp Sicheres FTP-artiges Programm auf SSH-Basis sftp(1) 35
ssh „Secure Shell“, erlaubt sichere interaktive Sitzungen auf anderen Rechnern ssh(1) 34
ssh-add Akkreditiert private Schlüssel beim ssh-agent ssh-add(1) 36
ssh-agent Verwaltet private Schlüssel und Kennwörter für die SSH ssh-agent(1) 36
ssh-keygen Generiert und verwaltet Schlüssel für die SSH ssh-keygen(1) 35, 45
sshd Server für das SSH-Protokoll (sicherer interaktiver Fernzugriff) sshd(8) 34

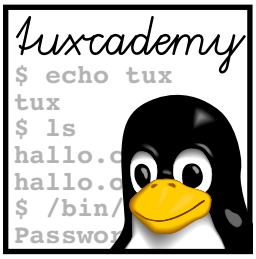
Zusammenfassung

- Die Secure Shell erlaubt das komfortable und sichere Anmelden auf entfernten Rechnern (und ersetzt so TELNET, RSH und RLOGIN) sowie die gesicherte Übertragung von Dateien ähnlich RCP und FTP.
- Mit OpenSSH steht eine leistungsfähige Implementierung der Secure Shell frei zur Verfügung.
- Zur Grundfunktionalität von SSH gehören die Ausführung einzelner Kommandos und ganzer interaktiver Sitzungen auf anderen Rechnern, das Kopieren von Dateien und die Weiterleitung von beliebigen TCP-Verbindungen über die verschlüsselte Strecke.
- SSH erlaubt eine Benutzerauthentisierung über Kennwörter (mit Verschlüsselung bei der Übertragung) oder über asymmetrische Schlüsselpaare.
- Mit `AllowUsers`, `DenyUsers`, `AllowGroups` und `DenyGroups` können Sie steuern, welche Benutzer sich über SSH anmelden dürfen. `PermitRootLogin` regelt den Zugriff von `root`.
- Der OpenSSH-Client (`ssh`) erlaubt eine detaillierte Konfiguration für verschiedene Gegenstellen.
- Die SSH-Protokollversion 1 sollten Sie aus Sicherheitsgründen nicht mehr benutzen.
- Für Konfigurationsänderungen am `sshd` empfiehlt es sich, parallel eine zweite Verbindung aufzubauen. Diese überlebt Neustarts des `sshd`.
- Öffentliche OpenSSH-Schlüssel können auf dem Server mit umfangreichen Optionen ausgestattet werden, die ihre Möglichkeiten einschränken.
- OpenSSH-Zertifikate erleichtern vor allem in großen Netzen die Administration von öffentlichen Benutzer- und Rechner-Schlüsseln

Literaturverzeichnis

BS01 Daniel J. Barrett, Richard Silverman. *SSH, The Secure Shell: The Definitive Guide*. Sebastopol, CA: O'Reilly & Associates, 2001. ISBN 0-596-00011-1. http://www.oreilly.com/catalog/sshtdg/

RFC4253 T. Ylonen, C. Lonvick. »The Secure Shell (SSH) Transport Layer Protocol«, Januar 2006. http://www.ietf.org/rfc/rfc4253.txt



4

Firewall-Konzepte

Inhalt

Lernziele

- Firewalls als Hilfsmittel zur Netzwerksicherheit kennen
- Firewall-Bestandteile wie Paketfilter, Application Level Gateways usw. kennen und einordnen können
- Firewall-Implementierungsmöglichkeiten kennen und Firewall-Infrastrukturen planen können
- Wichtige TCP/IP-Protokolle und ihr Zusammenwirken mit Firewall-Infrastrukturen bewerten können

Vorkenntnisse

- Allgemeine Kenntnisse über Rechner- und Netzwerksicherheit
- Netzwerk- und TCP/IP-Kenntnisse

4.1 Firewalls und Sicherheit

Firewalls, zu deutsch »Brandmauern«[1], stellen ein wichtiges technisches Hilfsmittel zu höherer Sicherheit in einem Rechnersystem dar. Bevor wir uns im Detail mit ihrer Struktur und Konzeption befassen, hier ein paar Hinweise darauf, was Firewalls *nicht* sind:

Ein Allheilmittel für jegliche Sicherheitsprobleme Firewalls können Bestandteil einer Sicherheitsinfrastruktur sein, aber nicht die ganze Sicherheitsinfrastruktur. Ihr Einsatz ist nie Selbstzweck, sondern ergibt sich aus einem Sicherheitskonzept, zu dem neben detaillierten Sicherheits-Zielvorgaben beispielsweise auch eine vernünftige restliche Infrastruktur, angemessene Protokollierung und nicht zuletzt adäquate personelle Ausstattung für die Administration gehören.

Ein käufliches Produkt Diverse Anbieter werden Ihnen mit Freuden Tausende von Euros für matt schimmernde Kisten abknöpfen, die Sie in Ihren 19-Zoll-Schrank schieben und an Ihren Internetanschluß hängen können. Die dadurch gewonnene Sicherheit ist minimal, wenn das Gerät nicht kompetent konfiguriert und administriert wird. Sie können also nicht erwarten, Netzwerksicherheit mit geringstmöglichem eigenen Aufwand in vorgekochter Form erwerben zu können. Auch hier wieder der Hinweis auf das Sicherheitskonzept, aus dem die Vorgaben für die Konfiguration eines Firewallsystems (egal ob fertig gekauft oder selbst etwa auf Linux-Basis zusammengesetzt) resultieren.

»Persönlich« Was man Ihnen als »persönlichen Firewall« verkauft – etwa für Windows oder openSUSE – ist in der Regel nichts als ein simpler Paketfilter (siehe Kapitel 5), der versucht, unautorisierte Verbindungen zu unterbinden, die zu Ihrem Rechner aufgebaut oder von Ihrem Rechner aus gestartet werden sollen. Mit den in diesem Kapitel besprochenen Infrastrukturen hat dies nur am Rande zu tun, und für Linux brauchen Sie so ein Produkt eigentlich auch nicht. Schaden kann es nicht (wenn man von der permanenten Belästigung durch aufpoppende Dialoge mal absieht), aber Sie sollten sich auch nicht allzuviel davon versprechen – hinreichend gemeine Programme können jedenfalls auf Systemen wie Windows durchaus Ihren »persönlichen Firewall« stoppen, bevor sie anfangen, Schindluder zu treiben.

Vertrauensgrenzen

Allgemein dienen Firewalls dazu, Netze (weitgehend) voneinander abzuschotten, denen man unterschiedlich weit vertraut. Das einfachste Beispiel ist hier ein Firewall, der zum Beispiel Ihr LAN daheim oder in der Firma vom Internet trennt. Im LAN haben Sie sehr weitgehende Kontrolle über die verwendete Hard- und Software, und Sie können auch die Benutzer notfalls Mores lehren; im Internet genießen Sie diese Privilegien nicht. Sie tun also gut daran, dem LAN mehr zu vertrauen (normalerweise nicht 100%) als dem Internet, und aus dieser Überlegung heraus rechtfertigt sich der Einsatz eines Firewalls an der Grenze zwischen LAN und Internet. (Wie dieser konkret aussieht, steht auf einem anderen Blatt.) Die genauen Vertrauensgrenzen ergeben sich aus Ihrem Sicherheitskonzept; so kann es durchaus möglich sein, dass innerhalb Ihrer Firma weitere Vertrauensgrenzen existieren, etwa zwischen dem allgemeinen Firmen-LAN und dem LAN der streng geheimen Entwicklungsabteilung.

Die Grenze zwischen zwei Netzen wie Ihrem LAN und dem Internet ist nicht notwendigerweise »scharf«: Es kann gut sein, dass Sie Serverdienste anbieten, die aus dem Internet zugänglich sind (Stichwörter: DNS, Web und Mail) und damit

[1] Im Bauwesen dient eine »Brandmauer« dazu, zu verhindern, dass ein Feuer auf einer Seite der Mauer schnell auf die andere Seite überspringt. Es handelt sich also um eine besonders feuerfest ausgelegte Wand, die die beiden Seiten effektiv voneinander trennen soll. Direkt auf Rechnernetze übertragen läßt sich das sehr einfach realisieren: Sie ziehen den Stecker aus der Buchse, und die Trennung ist erreicht. Natürlich ist das aber nicht das Ziel – Sie wollen kontrollierte Kommunikation zwischen den beiden Seiten zulassen. Und da fangen die Probleme an …

leichter angegriffen werden können als Arbeitsplatzrechner in Ihrem LAN, die vom Internet aus gar nicht sichtbar sein müssen. Darum ist es angebracht, diese Server mit mehr Argwohn zu betrachten als einen Arbeitsplatzrechner, und daraus folgt zumeist die Einrichtung einer »demilitarisierten Zone« (DMZ), eines »Niemandslands« zwischen LAN und Internet. Aus dem Internet sind Zugriffe auf Rechner in der DMZ möglich, aber keine Zugriffe auf Rechner im LAN. Umgekehrt können Sie genauso Internetzugriffe aus dem LAN nur über Rechner in der DMZ zulassen, etwa um den Web-Gebrauch zu protokollieren und zu steuern.

DMZ

Bedenken Sie auch, dass ein Firewall Sie im Idealfall vor Angreifern außerhalb Ihres lokalen Netzes schützt. Angreifer innerhalb Ihres LANs (verärgerte oder frustrierte Angestellte, externes Personal, …) müssen nicht über Ihren Firewall gehen, um auf Ihr internes Netz zuzugreifen, und alle Schutzmaßnahmen, die Sie dort vorsehen, sind ergo wirkungslos.

Interne Angreifer

4.2 Firewall-Bestandteile

Wie werden Firewalls aufgebaut? Bevor wir uns mit dieser Frage beschäftigen, zunächst einige Begriffe, mit denen wir zwangsläufig in Berührung kommen werden:

Paketfilter Paketfilter dienen dazu, den Verkehr zwischen zwei Netzen auf relativ niedriger Ebene zu inspizieren und zu kontrollieren. Sie operieren zumeist auf den ISO/OSI-Schichten 3 und 4; im Sinne von TCP/IP heißt das, dass auf der IP-Ebene und den darüberliegenden Transportprotokollen wie TCP, UDP oder ICMP gefiltert wird. In Filterregeln können Sie also auf Attribute Bezug nehmen wie die Absender- und Empfängeradresse eines Pakets, eventuell Absender- und Empfänger-Portnummern und andere Daten wie TCP-Flags oder ICMP-Pakettypen. Ferner sind natürlich auch die Netzschnittstelle interessant, über die ein Paket eingegangen ist, und (nach einer Routing-Entscheidung) die, über die es das System wieder verlassen soll. Leistungsfähige Paketfilter wie der »Netfilter« im Linux-Kern können außerdem feststellen, ob Pakete eine neue Verbindung aufbauen sollen oder Bestandteil einer bereits existierenden Verbindung sind.

Paketfilter werden gerne mit Routern kombiniert, die ohnehin eine Entscheidung über die Weiterleitung von Paketen treffen müssen.

Neben dem »klassischen« Paketfilter (`iptables`) unterstützt Linux einen Paketfilter auf ISO/OSI-Schicht-2-Ebene (`ebtables`), der nach außen aussieht wie eine Ethernet-Bridge – zwei Ethernet-Karten werden so miteinander verbunden, dass die daran angeschlossenen Netze aussehen wie ein einziges und zum Beispiel ohne Routing Adressen aus demselben IP-Subnetz verwenden können. Der Linux-Kern leitet Pakete zwischen den Netzen weiter und erlaubt dabei über eine `iptables`-Erweiterung deren detaillierte Inspektion mit allen Möglichkeiten von Netfilter. Auch das Filtern von ARP-Paketen ist möglich. Der Vorteil eines solchen Paketfilters (auch »Bridgewall« genannt) ist die völlige Transparenz gegenüber IP. Er läßt sich zum Beispiel in Netze integrieren, die aufgrund ihrer Struktur die Implementierung eines klassischen Schicht-3-Firewalls schwierig machen würden.

In dieser Unterlage werden Paketfilter für Linux im Detail in Kapitel 5 besprochen.

Application Level Gateways Im Gegensatz zu Paketfiltern arbeiten Application Level Gateways, wie ihr Name andeutet, auf der Ebene von Anwendungsprotokollen wie HTTP, FTP, SMTP oder DNS (ISO/OSI-Schicht 7). Man spricht auch von »Proxies«. Der Vorteil eines Application Level Gateways ist, dass es das Anwendungsprotokoll »mitlesen« kann und so einerseits eine detaillierte Protokollierung (Auditing) gestattet sowie andererseits Anfragen, die gegen das Sicherheitskonzept verstoßen, unterbinden kann. Wichtig für Application Level Gate-

ways ist natürlich, dass sie nicht umgangen werden können; wenn Sie beispielsweise vorhaben, Zugriffe aus Ihrem LAN auf das World-Wide Web über einen Application Level Gateway wie `squid` zu leiten, müssen Sie gleichzeitig dafür sorgen, dass Ihre LAN-Benutzer keine direkte Verbindung zu beliebigen Web-Servern im Internet aufbauen können. Dies läßt sich auf verschiedene Arten realisieren: Eventuell werden gar keine IP-Datagramme zwischen LAN und Internet geroutet, oder Zugriffe auf entsprechende Ports (typischerweise 80 für HTTP und 443 für HTTPS) werden im Router mit einem Paketfilter blockiert.

Wie gut ein Protokoll mit Application Level Gateways harmoniert, hängt vom Protokoll ab. HTTP, DNS und "*store-and-forward*"-Protokolle wie SMTP werfen kaum Probleme auf, während FTP Ihnen schon einiges Kopfzerbrechen bereiten kann. In manchen Protokolle wie HTTP werden Proxies explizit unterstützt, während ihre Existenz bei anderen (etwa DNS und SMTP) völlig transparent bleibt. Auch für HTTP können Sie übrigens »transparente Proxies« benutzen, um eine clientseitige Konfiguration zu vermeiden.

Bastion Hosts Ein "*bastion host*" (zu deutsch vielleicht »befestigter Rechner« – denken Sie an Mauern, Türme und Schießscharten) ist ein Rechner, der Dienste für ein weniger vertrauenswürdiges Netz, typischerweise das Internet, anbieten soll. Im Idealfall ist ein Bastion Host als Minimalsystem (Abschnitt 2.2) ausgelegt, enthält also genau die Software, die zur Erbringung *eines* Dienstes – etwa DNS oder Web – nötig ist, plus möglicherweise ein paar Hilfen zur Administration wie einen SSH-Daemon (fakultativ).

Minimalsystem

Daraus folgt unmittelbar, dass gängige allgemein verwendbare Linux-Distributionen (wie beispielsweise die von SUSE/Novell) zur Implementierung von *bastion hosts* nicht sehr taugen – zum Thema »Minimalsysteme« siehe Abschnitt 2.2).

Für den Einsatz als Router oder Firewall empfehlen sich speziell dafür ausgelegte Linux-Distributionen wie `fli4l` oder IPCop. Verwenden Sie dafür bitte keine Distribution vom Kaliber einer openSUSE, wenn Sie es irgendwie einrichten können.

Sie können Linux auch für einen Router oder Firewall verwenden, ohne dafür einen stromfressenden Standard-PC einsetzen zu müssen. Diverse preiswerte Router (ursprünglich etwa der WRT54GL von Linksys, aber auch viele neuere und leistungsfähigere Geräte) gestatten die Installation von speziellen Linux-Distributionen wie OpenWRT, deren Funktionsumfang in der Regel weit über die vom Hersteller gelieferte Firmware hinausgeht. Siehe hierzu etwa `http://www.openwrt.org/`.

Dual-Homed Hosts Ein "*dual-homed host*" ist ein Rechner mit zwei Netzwerkkarten, der *nicht* zwischen beiden routet. Jegliche Kommunikation zwischen den beiden Netzen erfolgt über Application Level Gateways. Der Vorteil dieses Ansatzes ist, dass Rechner im LAN nicht mit Paketen in Kontakt kommen können, die vom Internet kommen – sie reden nur mit dem *dual-homed host*. Böswillige Gestalten im Internet können Rechner im LAN also nicht mit plumpen paketbasierten Angriffen wie dem "*ping of death*"[2] attackieren, da sie aus dem Internet nicht direkt erreichbar sind. (Ein Paketfilter, der solche Sachen wegwirft, führt natürlich zum selben Effekt, aber bei einem *dual-homed host* ergibt sich das automatisch, ohne weitere Konfiguration.)

[2] Dieser Angriff nutzt aus, dass viele Betriebssysteme nicht prüfen, ob IP-Datagramme (insbesondere ICMP-Echo-Requests) die offizielle Maximalgröße von 65535 Bytes einhalten. Als der Angriff neu war (1996), konnte man diverse Rechner einfach durch ein Kommando wie »`ping -l 65510`« zum Absturz oder Neustart bringen. Wenn Sie noch irgendwo einen Windows-95-Rechner herumstehen haben, probieren Sie es aus – Microsoft hat das Problem in Windows 95 nie repariert (während ein Patch für Linux knapp drei Stunden nach Bekanntwerden des Problems zur Verfügung stand).

Adressierung, NAT und Portweiterleitung IP-Adressen sind heutzutage eine knappe Ressource, und bis zur flächendeckenden Einführung von IPv6 wird das auch noch so bleiben. Es ist also möglich, dass Sie für Ihr Netz nur eine oder ein paar IP-Adressen zur Verfügung haben – nicht genug für alle Serverdienste und erst recht nicht für alle Arbeitsplatzrechner. Die Abhilfe lautet hier Netzwerk-Adressumsetzung (engl. *network address translation* oder »NAT«) oder, genauer gesagt, einerseits Masquerading und andererseits Portweiterleitung. Der einzige Ihrer Rechner, der eine öffentliche IP-Adresse haben muss, ist der Router, der die Verbindung ins Internet herstellt. Alle anderen können private IP-Adressen nach [RFC1918] haben. Der Zugangsrouter führt für Internet-Zugriffe aus dem LAN bzw. der DMZ Masquerading durch, indem er die Absenderadresse der ins Internet gerichteten IP-Datagramme durch seine eigene ersetzt und die zurückkommenden Antwortdatagramme entsprechend an den eigentlichen Absender weiterleitet. Zugriffe aus dem Internet auf allgemein verfügbare Dienste (etwa DNS, Mail und HTTP) werden vom Router in ähnlicher Weise an die entsprechenden Ports der dafür zuständigen Server (in der DMZ) weitergeleitet. (Sie sollten der Versuchung widerstehen, Ihren Router gleichzeitig zum Web-, Mail- oder DNS-Server zu machen; die Gefahr, dass ein Angreifer über eines der großen dafür nötigen Serverprogramme Ihren Router kompromittiert und damit im schlimmsten Fall freien Zugriff auf Ihr lokales Netz bekommt, ist viel zu groß.)

Masquerading

Portweiterleitung

Übungen

4.1 [!2] Warum werden Paketfilter auf den Schichten 3 und 4 des ISO/OSI-Referenzmodells angesiedelt und nicht zum Beispiel auf der Schicht 2 (»Bridgewalling« ist ja eine Erweiterung)? Was würde verlorengehen, wenn ein Paketfilter nur auf Schicht 2 oder nur auf Schicht 3-Ebene arbeiten würde?

4.2 [3] Diskutieren Sie die Vor- und Nachteile von paketfilternden Routern im Vergleich zu *dual-homed hosts*.

4.3 [2] Im Kopf eines IP-Datagramms sind 16 Bit für die Datagrammlänge vorgesehen. Wie kann man also überhaupt ein Datagramm verschicken, das größer ist als 65535 Bytes?

4.4 [3] (Forschungsaufgabe.) Was ist SOCKS und wozu ist es gut? Wie können (nahezu) beliebige Linux-Programme SOCKS unterstützen?

4.3 Implementierung von Firewalls

4.3.1 Ein einfaches Beispiel: Heim-LAN

Eines der einfachsten Beispiele für eine »firewallartige« Netzstruktur findet sich im häuslichen Bereich: Ein Rechner mit Internetanschluss – etwa über ISDN oder DSL – erlaubt nicht nur dem direkt an ihm arbeitenden Benutzer den Zugriff auf E-Mail und das World-Wide Web, sondern ist auch in einem lokalen Netz (typischerweise auf der Basis von Ethernet oder drahtlosen LAN-Techniken) mit anderen Rechnern verbunden, etwa für Ehepartner oder Kinder, und vermittelt Internetzugriffe für diese. Diese Sorte Konfiguration wird von vielen populären Betriebssystemen, inklusive Linux, unterstützt, und ist eine naheliegende Erweiterung des simplen Falls eines direkt ans Internet angeschlossenen Einzelrechners.

Schon ein einzelner Rechner sollte nicht völlig gedankenlos ans Internet angeschlossen werden. Sie sollten in jedem Fall prüfen, dass der Rechner keine unerwünschten Netzdienste nach außen anbietet – bei einem Rechner, der mit einer Wählverbindung über einen ISP ins Netz geht (und das schließt eine DSL-Flatrate ein) sollten eigentlich gar keine Netzdienste aktiv sein, da die meisten sich sowieso nicht stabil zur Verfügung stellen lassen, wenn die IP-Adresse des Rechners sich

Wählverbindung über einen ISP

immer wieder ändert. (Einen Web-Server können Sie vermutlich anbieten, aber alles andere macht nicht wirklich Laune oder ist möglicherweise sogar gefährlich – ein Mailserver zum Beispiel sollte permanent unter derselben IP-Adresse erreichbar sein, sonst besteht die Gefahr von Mailverlust und anderen Arten von Verwirrung.)

Heutzutage können Sie davon ausgehen, dass neue direkt ans Internet angeschlossene Rechner binnen Minuten entdeckt und »gescannt« werden – normalerweise geht das schneller, als das System braucht, um sich die aktuellen Sicherheits-Updates aus dem Netz zu holen und sie zu installieren. Glücklicherweise werden heute nur wenige Rechner wirklich direkt ans Internet angeschlossen, sondern der Zugang erfolgt in der Regel über (DSL-) Router, deren rudimentäre Paketfilter-Funktionalität unaufgeforderte Zugriffe von außen verhindern kann. Das Gefahrenmoment wird dadurch um einiges verringert.

Unter Linux können Sie mit »`netstat -tul`« prüfen, welche Netzdienste nach außen angeboten werden. Seien Sie vorsichtig mit allem, was in der »Local Address«-Spalte ein `*` oder (allgemein) nicht nur »`127.0.0.1`« enthält. Wenn Sie genau wissen wollen, was auf Ihrem Rechner von außen zu sehen ist, dann schauen Sie von außen mit `nmap` (Abschnitt 6.2) nach.

Paketfilter

Ein Paketfilter ist für einen solchen Rechner nicht zwingend erforderlich. Sie sollten aber zur Sicherheit dafür sorgen, dass alle Verbindungsaufbauversuche von außerhalb (oder im Fall des Beispiels von außerhalb des LANs) per Paketfilter abgewiesen werden. Das hilft gegen versteckte »trojanische« Programme, die keinen Port öffnen, sondern zur Aktivierung auf Pakete lauschen, die eine bestimmte Sequenz von Zugriffen versuchen (»einmal auf Port 10000, einmal auf 11000, zweimal auf 12000« – sogenanntes *port knocking*). So ein Programm müßte unter Linux als `root` laufen und könnte sicher auch einen Paketfilter deaktivieren, aber je mehr es in den normalen Systembetrieb eingreift, desto eher können Sie es erkennen.

Ihr Internet-Rechner muss auch keine Pakete zwischen Ihrem LAN und dem Internet routen (siehe nächsten Abschnitt). Wenn es nur darum geht, Web-Zugriff zur Verfügung zu stellen, dann sollte auf dem Internet-Rechner ein Web-Proxy (beispielsweise Squid) laufen, der die Web-Ressourcen-Wünsche der Benutzer im LAN entgegennimmt und dann unter eigener Flagge die tatsächlichen Zugriffe durchführt. An dieser Stelle würden Sie auch ansetzen, wenn Sie beispielsweise Ihren minderjährigen Kindern nur Zugriff auf bestimmte Web-Seiten gestatten möchten: Der Web-Proxy kann jeden Zugriff gegen eine Liste erlaubter URLs prüfen und gegebenenfalls unterbinden, und wenn der Spielzimmer-PC keinen direkten Internetzugang hat und der Internet-Rechner hinreichend abgesichert ist, müssen Sie auch nicht unbedingt fürchten, von Ihren Sprößlingen ausgetrickst zu werden.

Web-Proxy

Mail

Entsprechend lohnt es sich vermutlich, Mail für alle Benutzer zentral mit dem Internet-Rechner abzurufen und dort einen POP- oder IMAP-Server zur Verfügung zu stellen. Dies erlaubt eine zentrale, von den verwendeten Mailprogrammen unabhängige Spam- und Virenfilterung und kann (falls nötig) Onlinezeit und/oder Transfervolumen sparen helfen. Ausgehende Mail können Sie über einen lokalen Mailserver auf dem Internet-Rechner leiten, der alle Nachrichten, die nicht an lokale Benutzer im Netz adressiert sind, an den Mailserver des ISP weiterreicht. Damit sind aus der Sicht eines Rechners im LAN die wesentlichen Dienste abgedeckt.

DNS

Auf DNS kann in einem solchen Netz entweder komplett verzichtet werden (zugunsten von `/etc/hosts`), oder Sie verwenden einen einfachen DNS-Server wie `dnsmasq` (`http://www.thekelleys.org.uk/dnsmasq/`), der die `/etc/hosts`-Datei auf dem Internet-Rechner per DNS im LAN zugänglich macht. BIND wäre an dieser Stelle vermutlich Overkill.

Tabelle 4.1: Eine einfache Kommunikationsmatrix

Hier steht »×« für »keine Kommunikation«, »+« für »beliebige Kommunikation«.

Von / Nach	LAN-Client	Internet-PC	Internet
LAN-Client	+	HTTP(S), SMTP, POP3, DNS	×
Internet-PC	+	+	HTTP(S), SMTP, POP3, DNS
Internet	×	×	(Nicht unser Problem)

Energie

Der Nachteil an dieser Stelle ist natürlich, dass der Internet-PC permanent eingeschaltet sein muss (oder doch zumindest dann, wenn andere Rechner Web-Zugriffe durchführen wollen). Je nachdem, wie dieser PC ausgerüstet ist, kann das implizieren, dass eine Menge Strom nutzlos verbraten wird. In diesem Fall ist es möglicherweise sinnvoll, als Internet-PC einen älteren Rechner mit minimaler zusätzlicher Hardware (keine leistungsstarke, aber wärmeproduzierende Grafikkarte usw.) zu verwenden oder – bei entsprechender Brieftasche – eines der neuen passiv gekühlten »Barebone«-Systeme.

4.3.2 Ein Heim-LAN mit Router

Die nächste Steigerung ist eine Konfiguration, in der der Internet-PC nicht nur als Web-Proxy, sondern als Router fungiert, der direkt Pakete aus dem LAN ins Internet (und zurück) weiterreicht. Dabei wird mit relativ großer Sicherheit Masquerading zum Einsatz kommen, um im LAN die Verwendung von privaten IP-Adressen nach [RFC1918] zu ermöglichen. Dienste wie Web (HTTP/HTTPS), Mail und DNS sollten Sie aber am besten wie im vorigen Beispiel über Proxies laufen lassen.

Paketfilter

Im Gegensatz zum vorigen Beispiel, wo keine direkte Verbindung zwischen LAN und Internet bestand, ist hier ein geeignet konfigurierter Paketfilter unverzichtbar. Prüfen Sie, welche Internet-Dienste vom Internet-PC und von LAN-Clients aus direkt zugänglich sein sollen und welche über Proxies geleitet werden. Aus dieser »Kommunikationsmatrix« (Tabelle 4.1) ergibt sich die Paketfilterkonfiguration.

Kommunikationsmatrix

Internetdienste

Sie sollten es sich auch in dieser Konfiguration tunlichst verkneifen, Internetdienste nach draußen anzubieten. Es wäre möglich, etwa einen Web-Server auf einem Rechner im LAN zu installieren und den Port 80 des Routers auf diesen Rechner weiterzuleiten. Dies verschafft aber Internet-Benutzern Zugang zu einem Rechner in Ihrem LAN, und wenn es einem Cracker gelingt, Ihren Webserver-PC zu kompromittieren, steht ihm sofort Ihr komplettes lokales Netz offen. Um Dienste nach außen anzubieten und dennoch Ihr LAN vor direktem Zugriff zu schützen, brauchen Sie eine Konfiguration mit DMZ.

4.3.3 Internet-Anbindung einer Firma mit DMZ

Als Systemverwalter in einem (typischerweise mittleren bis großen) Unternehmen möchten Sie mitunter diverse Dienste selbst in die Hand nehmen, die Privatleute und kleine Firmen gerne einem ISP überlassen, etwa DNS, einen Webserver und das Annehmen von Mail über SMTP. Dazu brauchen Sie zunächst eine Standleitung (oder deren moralisches Äquivalent) sowie eine oder mehrere feste IP-Adressen, damit Sie Ihre Dienste im internetweiten DNS ausloben können.

feste IP-Adressen

Die gängigen DSL-Zugänge für Privathaushalte eignen sich nicht dafür, Dienste anzubieten, da die zugeordnete IP-Adresse (typischerweise) alle 24 Stunden wechselt. Dynamische DNS-Dienste, können das bis zu einem gewissen Grad abfedern – zumindest für Protokolle wie HTTP –, aber zum Beispiel einen SMTP-Server können Sie so nicht stabil betreiben.

Wichtig ist zunächst, dass der oder die Rechner, die DNS, Web oder SMTP anbieten (die *bastion hosts*), nicht in Ihrem lokalen Netz aufgestellt werden, sondern in der »demilitarisierten Zone« (DMZ) zwischen LAN und Internet. Paketfilternde Router stellen die Verbindung zwischen DMZ und Internet auf der einen und DMZ und LAN auf der anderen Seite her; sie werden so konfiguriert, dass aus dem Internet heraus kein direkter Zugang ins LAN möglich ist, aber die *bastion hosts* in der DMZ direkt oder über eine Portweiterleitung am internetseitigen Router erreicht werden können. Umgekehrt sind aus dem LAN Zugriffe auf Proxies für Web und Mail möglich (die sinnvollerweise auch in der DMZ untergebracht werden); diese Proxies haben dann Zugang zum Internet, aber zwischen LAN und Internet besteht wieder keine direkte Verbindung mehr.

Vorteile

Diese Art von Konfiguration hat verschiedene Vorteile:

- Angreifer aus dem Internet können nur versuchen, die *bastion hosts* in der DMZ zu kompromittieren. Gelingt ihnen das, haben sie zwar Zugriff auf andere Systeme in der DMZ (was ärgerlich genug ist), aber die Rechner im LAN sind nach wie vor sicher.

- Im Idealfall steht für jeden Dienst in der DMZ ein eigener *bastion host* zur Verfügung, der speziell dafür als Minimalsystem ausgelegt ist. Ein Angreifer, der einen Dienst kompromittiert, findet ein System vor, das auf das Allernötigste abgespeckt wurde, und kann nicht direkt weitere Dienste ins Visier nehmen, die auf demselben Rechner laufen.

  Virtualisierung macht es heute möglich, nur einen physikalischen Rechner zu betreiben, auf diesem aber mehrere »virtuelle« Server laufen zu lassen, die unterschiedliche Dienste jeweils voneinander abgeschottet erbringen. Dies ist eine extrem interessante Entwicklung, die in dieser Schulungsunterlage leider nicht behandelt werden kann.

- Die paketfilternden Router an den Enden der DMZ können sehr simpel ausgelegt sein, im Vergleich zum »Internet-PC« der vorigen Beispiele, der möglicherweise noch diverse andere Rollen zu erfüllen hatte. Es muss sich dabei auch nicht (oder nicht nur) um Linux-Rechner handeln, auch wenn das das ist, was uns hier am meisten beschäftigen wird; unter Umständen ist es sogar vorteilhaft, hier einen Linux-Rechner und einen hardwarebasierten Firewall-Router oder Firewall-Router verschiedener Hersteller einzusetzen, damit im Falle von Implementierungsfehlern im Paketfilter nach wie vor ein gewisser Schutz für das LAN gegeben ist und ein Angreifer nicht mit denselben Tricks beide Paketfilter durchbrechen kann[3].

- In der DMZ können weitere Diagnose- und Beobachtungswerkzeuge zum Einsatz kommen, die beispielsweise Angriffe, die den Internet-Router durchbrochen haben, zu erkennen und zu melden versuchen. Solche Werkzeuge können so ausgelegt werden, dass sie für einen Angreifer nicht zu finden und zu kompromittieren sind.

Nachteil

Als Nachteil steht diesen zweifellos interessanten und wünschenswerten Vorteilen vor allem der sehr hohe Hardware- und Administrationsaufwand gegenüber, der dazu führt, dass diese Sorte Konfiguration, konsequent bis zu Ende gedacht, nur für Installationen mit beträchtlichem Budget sowohl für Rechner- und Netzwerkinfrastruktur als auch für Personal in Frage kommt. In der wirklichen Welt werden oft Vereinfachungen vorgenommen wie die, mehrere DMZ-Dienste auf demselben *bastion host* zu implementieren. Eine solche Infrastruktur ist potentiell immer noch viel sicherer als eine ohne DMZ, aber nicht mehr optimal.

[3] Dies beruht auf der Annahme, dass die Routerhersteller nicht dieselben Fehler machen wie die Linux-Programmierer. Wenn Ihnen das wichtig ist, sollten Sie aufpassen, keinen Router zu erwischen, der wie diverse heute am Markt erhältliche Produkte intern auf Linux basiert.

4.3.4 DMZ für Arme: Triple-Homed Host

Eine weitere Vereinfachung für Installationen mit geringem Budget verzichtet auf die Kombination von zwei paketfilternden Routern an den beiden Enden der DMZ und ersetzt diese durch einen Rechner mit drei Netzwerkkarten – je eine für die Internetanbindung, das LAN und die DMZ. Auch das kann funktionieren, aber die Grundbedingung ist hier ein sorgfältig konfigurierter Paketfilter, der Zugriffe vom Internet ins LAN unterbindet, vom LAN ins Internet größtenteils unterbindet (und nur unter ganz kontrollierten Umständen erlaubt) und die Kommunikation zwischen Internet und DMZ sowie DMZ und LAN genau steuert. Entwerfen Sie auch hier eine Kommunikationsmatrix, um den Überblick über die Paketfilterkonfiguration zu behalten.

Die auf Firewalls und Router spezialisierte Linux-Distribution IPCop (`http://www.ipcop.org/`) unterstützt auch diese Sorte Konfiguration. Dasselbe gilt für dedizierte Router mit Distributionen wie OpenWRT.

Übungen

4.5 [!3] Herr Schulz möchte sein Heim-LAN neu organisieren. Er selbst hat einen recht neuen PC, und seine beiden Töchter haben das Vorgängerexemplar »geerbt«. Außerdem gibt es einen älteren Rechner, der – mit zwei Ethernet-Karten vom Discounter an der Kasse – als Internet-Zugangsrechner dienen soll. Herr Schulz möchte im Web surfen, E-Mail vom Provider abrufen (per POP3) und senden (mit SMTP zum Provider); außerdem nutzt er IRC und gelegentlich Server für internetbasierte Multiplayer-Spiele. Seine Töchter bekommen nur E-Mail und – zu bestimmten Tageszeiten – Web-Zugriff. – Stellen Sie eine Kommunikationsmatrix für dieses Szenario auf. Welche Benutzer gibt es? Welche Dienste werden wie verwendet? Welche Daten fließen von wo nach wo?

4.6 [2] Schon billige Internet-Zugangsrouter bieten meist eine Möglichkeit an, Ports am Router auf Rechner im angebundenen Netz weiterzuleiten und so Dienste im Internet anzubieten. Was halten Sie davon?

4.7 [3] Diskutieren Sie im Detail die Sicherheitskompromisse eines *triple-homed hosts* gegenüber einer vollständigen Firewall-Infrastruktur mit DMZ.

4.4 Firewalls und gängige Protokolle

Zum Abschluss dieses Kapitels hier einige Anmerkungen zu gängigen Protokollen aus der Sicht eines Firewall-Administrators. Die Protokolle selbst werden hier nicht im Detail erklärt; greifen Sie gegebenenfalls auf die Dokumentation (RFCs) oder andere Linup-Front-Schulungsunterlagen zurück.

HTTP Das Web-Protokoll ist wegen seiner extrem simplen Struktur recht »firewallfreundlich«. Um HTTP-Zugriffe durch einen Paketfilter zu lassen, müssen Sie im Grunde nur solche Verbindungen erlauben, die von einem »hohen« TCP-Port (Portnummer 1024 oder mehr) an einen Rechner im LAN zum TCP-Port 80 auf einem Rechner im Internet gerichtet sind.

Manche Web-Server verwenden andere Portnummern, etwa 8000, 8080 oder (selten) 81. Sie müssen sich überlegen, ob Sie solche Zugriffe zulassen möchten; möglicherweise handelt es sich dabei um inoffizielle Server, die schädlichen Inhalt anbieten, und möglicherweise auch nicht.

Im wirklichen Leben ist es aber oft sinnvoller, HTTP nicht direkt vom Client im LAN zum Server im Internet durchzulassen, sondern es über einen Proxy-Ser- Proxy-Server

ver (typischerweise Squid, aber es gibt andere) zu leiten. Der Proxy wird im Web-Browser des Clients eingetragen und bekommt dann von diesem alle Anfragen vorgelegt, die der Client sonst direkt an Server im Internet stellen würde; der Proxy prüft diese Anfragen, führt sie gegebenenfalls selbst (im eigenen Namen) durch und reicht die erhaltenen Antworten dann als HTTP-Resultate an den Client weiter. Neben der kompletten Entkopplung von LAN und Internet hat dieser Ansatz den Vorteil einer genauen Kontroll- und Protokollierungsmöglichkeit der Web-Zugriffe von Clients sowie der möglichen Zwischenspeicherung (engl. *caching*) von Inhalten, um künftige Anfragen nach derselben Ressource zu beschleunigen und Netzlast und/oder Kosten zu verringern.

Transparente Proxies

Neben »expliziten«, im Browser einzutragenden Proxies werden für HTTP auch »transparente« Proxies eingesetzt. Bei einem transparenten Proxy dient ein Paketfilter mit Adressumsetzung dazu, Zugriffe auf *irgendeinen* Web-Server im Netz auf den Proxy umzuleiten, der dann den tatsächlichen Zugriff durchführt und die Antwort so weiterleitet, dass der Web-Client annimmt, er hätte direkt mit dem eigentlichen Server gesprochen. Transparente Proxies haben den Vorteil, dass eine gezielte Konfiguration der Browser auf den Clients nicht notwendig ist – in einem Umfeld mit vielen verschiedenen Browsern oder mit nicht proxyfähiger HTTP-Client-Software kann das nützlich sein.

HTTPS Hierbei handelt es sich um HTTP, das mit SSL bzw. TLS für die Übertragung verschlüsselt wurde. HTTPS verwendet den TCP-Port 443 und kann in einem Paketfilter ansonsten genauso behandelt werden wie HTTP.

HTTPS und Proxies

Problematisch wird es mit HTTPS und Proxies. Einer der wesentlichen Vorteile von HTTPS ist die Authentisierung des Servers gegenüber dem Client, und genau diese ist nicht gegeben, wenn keine direkte (verschlüsselte) Verbindung zwischen Client und Server besteht – stellen Sie sich vor, ein Angreifer hätte Ihren Proxy kompromittiert und könnte jetzt Ihre HTTPS-Transaktionen (Online-Banking, E-Commerce, …) mitverfolgen. Ein Web-Proxy kann für HTTPS also nicht so funktionieren wie für HTTP, nämlich indem er Anfragen vom Client entgegennimmt und unter seiner eigenen Identität neu an den Server stellt, denn der Client würde nicht mehr den Zielserver als Gegenstelle sehen, sondern nur den Web-Proxy. Statt dessen unterstützen Proxies wie Squid HTTPS-Zugriffe, indem sie die vom Client geschickten verschlüsselten Bytes lesen und über eine neue TCP-Verbindung an den Zielserver weiterleiten. Die vom Server geschickten verschlüsselten Bytes werden entsprechend an den Client zurückgeschickt. Die Trennung von LAN und Internet ist damit weiterhin gegeben, aber der Proxy kann keine Kenntnis von den Kommunikationsinhalten nehmen, während der Client tatsächlich das Zertifikat des Servers übertragen bekommt. Damit das funktioniert, muss der Client zunächst den Proxy anweisen, die Byte-Weiterleitung einzurichten.

Transparente Proxies

Transparente Proxies sind für HTTPS nicht möglich, da der Client nicht weiß, dass ein Proxy im Spiel ist, und entsprechend keine besonderen Vorkehrungen treffen kann, um eine Byte-Weiterleitung anzufordern. Der Proxy könnte zwar die Bytes an den entfernten Server schicken, aber Client und Server würden als erstes versuchen, eine verschlüsselte Verbindung aushandeln, die der Proxy sowieso nicht mehr mitlesen kann, und die IP-Absenderadresse der vom Proxy an den Client zurückgeschickten Datagramme würde nicht mehr zum Zertifikat des Servers passen, so dass der Browser auf dem Client protestiert.

Es gibt Firewall-Produkte, die dennoch einen transparenten Proxy für HTTPS implementieren. Das funktioniert in der Regel so, dass der Proxy bei Bedarf ein Zertifikat generiert, das dem Client die richtigen Daten für den entfernten Server vorspielt[4]. Dabei geht natürlich jegliche Sicherheit flöten, die der Benutzer sich von dem HTTPS-Zugriff verspricht; der Vorteil ist hier aus der Sicht des Proxy-Betreibers wieder die Kontrolle und

[4] Das funktioniert, weil SSL/TLS dummerweise jeder beliebigen Zertifizierungsstelle erlaubt, ein Zertifikat für jeden beliebigen Rechnernamen auszustellen – man könnte das mit einiger Berechtigung als Entwurfsfehler ansehen.

Protokollierung von Zugriffen. Diese Sorte Proxy finden Sie (außer in totalitären Staaten) natürlich nicht bei Internet-Providern, sondern eher in großen Firmen, die versuchen, ein Auge auf die Internet-Aktivitäten ihrer Angestellten zu haben.

Voraussetzung dafür, dass das klappt, ist, dass die Browser das vom Firewall erzeugte Zertifikat unhinterfragt als »echt« akzeptieren. Das heißt normalerweise, dass das Zertifikat für die »Zertifizierungsstelle« des Firewalls im Browser installiert sein muss – was eine Firma, die die Softwareausstattung ihrer PCs kontrolliert, natürlich problemlos erreichen kann. (Die totalitären Staaten haben es da etwas schwerer; sie brauchen die – freiwillige, erzwungene oder unbemerkte – Kooperation einer »echten«, bei den gängigen Browsern akkreditierten Zertifizierungsstelle. Browserhersteller sind von solchen Machenschaften, wenn sie sie bemerken, normalerweise weniger als begeistert.)

Wie bei allen derartig dreisten und in Rechtsgüter von Verfassungsrang eingreifenden Überwachungsmaßnahmen sollten Sie in Ihrer Firma so etwas natürlich nur installieren (lassen), wenn das Management, die Personalvertretung und die Rechtsabteilung es abgesegnet haben und entsprechende Betriebsvereinbarungen existieren.

SMTP Das Mail-Übertragungsprotokoll gehört zur Klasse der *store-and-forward*-Protokolle; Nachrichten werden von Server zu Server weitergeleitet, und ein Server, der eine Nachricht annimmt, akzeptiert die Verantwortung für die weitere Zustellung (in Postfächer oder an andere Server, SMTP oder nicht). SMTP verwendet den TCP-Port 25 und kann an einem Paketfilter ähnlich behandelt werden wie oben bei HTTP beschrieben; für eingehendes SMTP ist analog der TCP-Port 25 auf der IP-Adresse des Mailservers für Gegenstellen mit beliebigen IP-Adressen und hohen Ports freizuschalten.

store-and-forward

Eine Bedrohung stellen heutzutage (Windows-)PCs von unbedarften Benutzern dar, auf denen »trojanische« SMTP-Server laufen und tonnenweise Spam verschicken. In immer mehr Netzen, vor allem bei kommerziellen Einwahl-Providern, wird darum ausgehendes SMTP an Port 25 auf beliebigen Zieladressen nicht mehr durchgelassen, damit die gecrackten Rechner nicht das ganze Netz in Verruf bringen können. Statt dessen erlauben diese Netze die Einlieferung von SMTP-Mail an einen lokalen Server, der diese Nachrichten dann weiterleitet. Hierzu wird der TCP-Port 587 verwendet [RFC2476], etwa mit Authentisierung [RFC2554] und einer Begrenzung auf eine bestimmte Nachrichtenanzahl und Datenmenge pro Zeiteinheit, um den Schaden für den Fall zu begrenzen, dass ein trojanisches Programm sich die SMTP-Authentisierungsinformationen des Benutzers aneignet.

TLS wird bei SMTP heute typischerweise »im Protokoll« zur Verfügung gestellt, indem ein (E)SMTP-Server bekanntgibt, dass er das `STARTTLS`-Kommando unterstützt, mit dem der Client darum bitten kann, dass die existierende SMTP-Verbindung auf »SMTP über TLS« aufgestockt wird [RFC3207].

TLS

Ebenfalls theoretisch möglich ist SMTPS (oder SSMTP), das sich zu SMTP verhält wie HTTPS zu HTTP und gesondert behandelt werden muss. Früher war diesem Dienst der Port 465 zugeordnet. Ältere Outlook-Versionen bestehen auf dessen Verfügbarkeit, offizell ist es aber längst nicht mehr.

In Infrastrukturen mit einer DMZ werden LAN und Internet meist durch einen SMTP-Proxy in der DMZ entkoppelt, der aus dem Internet zu erreichen ist und Nachrichten für die lokale Domain annimmt. Dieser SMTP-Proxy kann Spam- und Virenfilterung übernehmen und leitet akzeptable Nachrichten dann an einen SMTP-Server im LAN weiter, der sie an Benutzerpostfächer zustellt; der Paketfilter zwischen DMZ und LAN sollte SMTP-Datenverkehr nur zwischen diesen beiden Rechnern akzeptieren. Umgekehrt senden Clients im LAN ihre Nachrichten

SMTP-Proxy

an den SMTP-Server im LAN, der nichtlokale Nachrichten an den SMTP-Proxy in der DMZ weiterleitet. Dieser wiederum stellt sie entweder direkt zu (an einen laut MX-Datensatz für die Ziel-Domain zuständigen SMTP-Server) oder übergibt sie dem SMTP-Server des ISP für die weitere Zustellung.

SMTP-Server

Alle gängigen SMTP-Server (Postfix, Sendmail, Exim, Qmail, …) lassen sich als SMTP-Proxies konfigurieren. Eine detaillierte Diskussion der entsprechenden Konfigurationen für Postfix und Sendmail finden Sie in der Linup-Front-Schulungsunterlage *Linux als Mailserver*.

DNS Ähnlich wie bei SMTP sind die meisten DNS-Server auch als »Proxies« zu gebrauchen. Installieren Sie in Ihrem LAN einen DNS-Server, der alle Anfragen über nichtlokale Namen an einen DNS-Proxy in der DMZ weiterreicht; dieser kümmert sich entweder selbst um die rekursive Auflösung dieser Anfragen oder gibt sie wiederum weiter an den oder die DNS-Server Ihres ISP (was durch deren Caching Vorteile bringen kann und eine weitere Einschränkung am Paketfilter zwischen DMZ und Internet zuläßt). Anfragen aus dem Internet sollte ebenfalls ein DNS-Server in der DMZ beantworten.

DNS und Firewalls

Es ist am besten, wenn der »rekursive« DNS-Proxy in der DMZ, der sich um die Auflösung von Anfragen aus dem LAN kümmert, und der »autoritative« DNS-Server, der dem Internet Informationen über die nach außen sichtbaren Namen Ihrer Installation zur Verfügung stellt, nichts miteinander zu tun haben (wir sprechen von einer »geteilten DNS-Konfiguration« oder *split DNS configuration*). Insbesondere sollte der autoritative Server *nur* Anfragen nach lokalen Namen beantworten und *nicht* nebenberuflich auch noch rekursive Anfragen auflösen. Dies hilft gegen Angriffe wie *DNS cache poisoning*, bei dem der Cache eines rekursiven Servers durch gezielte Anfragen aus dem Internet mit falschen Informationen »vergiftet« wird, so dass Browser im betroffenen Netz beim Zugriff auf eigentlich unverfängliche URLs wie `http://www.beutelschneider-bank.de/login` an andere Server weitergeleitet werden, die täuschend ähnliche Seiten liefern, aber den Anwendern Anmeldeinformationen zur späteren kriminellen Verwendung entlocken (Stichwort »Phishing«)[5].

DNS cache poisoning

Natürlich sollte Ihr autoritativer DNS-Server in der DMZ nur Namen wie `www.example.com` oder `mail.example.com` und vielleicht einen MX-Datensatz für `example.com` im Internet verfügbar machen. Die Namen der Rechner in Ihrem LAN gehen die Welt nichts an! Details der Konfiguration des verbreitetsten DNS-Servers, BIND, finden Sie in den Linup-Front-Schulungsunterlagen *DNS und BIND* oder *Linux-Netzadministration*.

FTP Das ehrwürdige FTP-Protokoll ist in seiner normalen Form extrem firewallunfreundlich: Beim »aktiven FTP« baut der FTP-Client eine »Steuerverbindung« (engl. *control connection*) zum TCP-Port 21 des FTP-Servers auf, um diesem Kommandos zu schicken. Wird eine Datenübertragung notwendig (etwa um Dateien zu lesen oder auch nur eine Dateiliste), so schlägt der FTP-Client dem Server einen beliebig wählbaren Port vor, zu dem der FTP-Server dann eine TCP-Verbindung aufbaut, um die Daten zu schicken. Aus der Sicht eines Firewall-Administrators ist das natürlich ein Alptraum, denn genau dieses Szenario – ein Rechner »von außen« darf Verbindungen zu beliebigen Ports auf einem Rechner »innen« aufbauen – soll die Firewall-Infrastruktur ja gerade verhindern. Abhilfe verspricht das »passive FTP«, bei dem der Server dem Client einen Port für die Datenverbindung vorschlägt und der Client die Verbindung aufbaut; mit beliebigen Verbindungen von innen nach außen kann man sich notfalls noch eher anfreunden als umgekehrt.

Passives FTP löst das Problem natürlich nicht, sondern verschiebt es nur von der Client- auf die Serverseite (wo ja möglicherweise auch ein Firewall erwünscht ist). Gute FTP-Server erlauben es, einen Bereich von Portnummern

[5] Die extrem verbreitete Ignoranz von Web-Nutzern gegenüber HTTPS und SSL sorgt dafür, dass auch HTTPS mitunter nicht gegen solche plumpen Angriffe hilft.

zu konfigurieren, aus dem der Server sich dann für passives FTP bedient. Diese Portnummern können dann am Paketfilter freigeschaltet werden.

content inspection

Das Mittel der Wahl zur Paketfilterung bei FTP ist inzwischen *content inspection* bzw. *stateful filtering* (zwei Namen für dasselbe Konzept). Hierbei liest der Paketfilter das FTP-Protokoll mit und stellt fest, wenn bei aktivem FTP der Client dem Server einen Port vorschlägt. Dieser Port wird dann für eine Weile für Verbindungen vom TCP-Port 20 (`ftp-data`) des FTP-Servers aus freigeschaltet und nach dem Ende der Datenverbindung wieder gesperrt. Für Puristen verletzt ein solcher Paketfilter das Axiom, dass Paketfilter auf den ISO/OSI-Schichten 3 und 4 agieren (FTP lebt in Schicht 7), aber pragmatisch gesehen ist das eine sehr praktische Sache. Der Linux-Paketfilter, Netfilter, beherrscht das natürlich.

Proxies für FTP

Proxies für FTP sind nicht so verbreitet wie die für HTTP. Der verbreitete HTTP-Proxy Squid kann auch Ressourcen von FTP-Servern holen, aber spricht auf der Clientseite nur HTTP, nicht FTP. »Echte« FTP-Proxies gibt es eigentlich kaum, da FTP als Protokoll im Gegensatz zu HTTP nichts über Proxies weiß. Eine Proxy-Infrastruktur für FTP muss demnach auf transparenten Proxies beruhen; die »SUSE-Proxy-Suite« (GPL; `http://www.suse.com/en/whitepapers/proxy_suite/`) besteht im wesentlichen aus einem FTP-Proxy, mit dem – unter Schützenhilfe vom Paketfilter – auch kompliziertere Konfigurationen realisiert werden können, etwa »Zugang aus dem Internet für Download von einem FTP-Server in der DMZ; Zugang aus dem LAN für Download von beliebigen FTP-Servern im Internet; Zugang aus dem LAN für Upload und Download zu und von dem FTP-Server in der DMZ«. Allerdings entscheiden die meisten Installationen sich auch für solche Anwendungen doch eher für eine Kombination aus Squid (für den Zugang zu entfernten FTP-Servern) und `rsync` und/oder SSH zum Schreiben auf den eigenen FTP-Server in der DMZ.

Auch bei der SUSE scheint die Proxy-Suite kein Projekt von besonderer Priorität zu sein; die letzte (unvollständige) Version datiert von 2005. Solche Open-Source-Software ist entweder perfekt oder tot.

NFS und NIS Die RPC-basierten Protokolle eignen sich nicht wirklich für den Einsatz mit Paketfiltern, da die Portnummern der Server von Fall zu Fall wechseln können. Da die Protokolle aber ohnehin keine Anstrengungen unternehmen, in irgendeiner Weise sicher zu sein, sollten Sie *dringend* der Versuchung widerstehen, einen NFS- oder gar NIS-Server vom Internet aus zugänglich zu machen. Wenn Sie Dateiserverzugriff über das Internet benötigen – etwa um Heimarbeiter anzubinden –, dann tun Sie das bitte, bitte über ein VPN (Kapitel 9).

NetBIOS, SMB, CIFS (oder wie das Protokoll diese Woche heißt) Das im vorigen Abschnitt über NFS und NIS Gesagte gilt sinngemäß auch für den Zugriff auf Windows-artige Dateiserver (oder Linux-Rechner, die sich mit Samba als solche ausgeben). Sie tun gut daran, die von »NetBIOS über TCP/IP« benutzten TCP- und UDP-Ports 137 bis 139 am Paketfilter zu sperren, da nicht nur die Protokolle zum Missbrauch geradezu einladen, sondern auch plumpe *Denial-of-Service*-Angriffe auf diese Ports populär sind. Wie auch bei NFS sollten Sie entfernte Dateizugriffe auf einen Windows- oder Samba-Server nur über ein VPN erlauben.

Neuere Windows-Systeme benutzen NetBIOS-loses SMB, das direkt auf TCP/IP aufsetzt und nur noch den Port 445 (TCP und UDP) benötigt. Das ändert aber nichts daran, dass SMB-Server nicht ans Internet gehören.

Übungen

4.8 [!3] Studieren Sie POP3 [RFC1939] und erklären Sie, wie Sie einen Paketfilter so konfigurieren würden, dass er Rechnern im LAN den Zugriff auf POP3-Server im Internet erlaubt. Was sagen Sie zum Thema »POP3-Proxies«?

Zusammenfassung

- Firewalls dienen zur Trennung von Netzen an Vertrauensgrenzen, etwa zwischen LAN und Internet.
- Firewalls sind kein fertiges Produkt, sondern Bestandteil einer Sicherheitsinfrastruktur auf der Basis eines Sicherheitskonzepts.
- Eine »demilitarisierte Zone« (DMZ) befindet sich (zum Beispiel) zwischen LAN und Internet und beherbergt Server, die zum Internet sichtbar sein müssen, von denen aus aber kein LAN-Zugriff möglich sein soll.
- Firewalls schützen nur vor Angreifern außerhalb des vertrauenswürdigen Netzes.
- Paketfilter und Application Level Gateways (Proxies) sind wichtige Bestandteile von Firewall-Infrastrukturen.
- Auch Einzelrechner sollten nicht ohne Vorbedacht ans Internet angeschlossen werden.
- Einfache Paketfilter können Einzelrechner oder LANs vor unbefugten Zugriffen aus dem Internet schützen, solange keine Dienste nach außen angeboten werden.
- Eine komplette Firewall-Infrastruktur mit DMZ bietet die größte Sicherheit, aber verlangt auch den höchsten Aufwand für Hardware und Administration. Kompromisse sind oft notwendig.
- Ein *triple-homed host* kann eine kostengünstigere Alternative zu einer kompletten Firewall-Infrastruktur darstellen. Diese Konfiguration wird von fertigen Linux-Distributionen wie IPCop unterstützt.
- Populäre TCP/IP-basierte Protokolle sind gut (HTTP, SMTP, …) oder weniger gut (FTP, NFS, NIS, …) für die Verwendung mit Firewall-Infrastrukturen geeignet.

Literaturverzeichnis

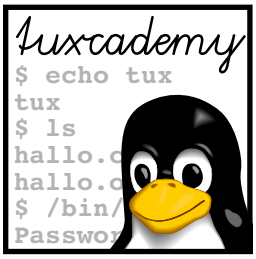
RFC1918 Y. Rekhter, B. Moskowitz, D. Karrenberg, et al. »Address Allocation for Private Internets«, Februar 1996. http://www.ietf.org/rfc/rfc1918.txt

RFC1939 J. Myers, M. Rose. »Post Office Protocol – Version 3«, Mai 1996. http://www.ietf.org/rfc/rfc1939.txt

RFC2476 R. Gellens, J. Klensin. »Message Submission«, Dezember 1998. http://www.ietf.org/rfc/rfc2476.txt

RFC2554 J. Myers. »SMTP Service Extension for Authentication«, Mârz 1999. http://www.ietf.org/rfc/rfc2554.txt

RFC3207 P. Hoffman. »SMTP Service Extension for Secure SMTP over Transport Layer Security«, Februar 2002. http://www.ietf.org/rfc/rfc3207.txt



5

Paketfilter mit Netfilter (»iptables«)

Inhalt

Lernziele

- Die Bedeutung von Paketfiltern verstehen
- Die Eigenschaften und Einsatzmöglichkeiten des Linux-Paketfilters Netfilter kennen
- Netfilter konfigurieren können

Vorkenntnisse

- TCP/IP-Kenntnisse
- Linux-Netzwerkkonfiguration
- Routing

5.1 Sinn und Zweck von Paketfiltern

Eine Anbindung eines einzelnen Rechners oder eines ganzen Netzes ans Internet ohne den Schutz durch eine Firewall-Infrastruktur wird heutzutage hoffentlich niemand mehr ernsthaft in Erwägung ziehen. Die stetig wachsende Verbreitung von Linux auf Intel-basierten PCs und die damit verbundene Entwicklergemeinde hat dafür gesorgt, dass mittlerweile nahezu alle Arten von Anwendungen unter Linux verfügbar sind. Dazu zählt auch ein im Kernel integrierter Paketfilter-Mechanismus. Ein Paketfilter untersucht die Header von IP-Datagrammen und entscheidet anhand von Regelketten über deren weiteres Schicksal. So können Sie mit Linux-Paketfilter Linux-basierten PCs eine sehr preiswerte Firewall-Lösung aufbauen. Die Funktionen, die ein solcher Linux-Firewall bietet, umfassen unter anderem:

- IP-Filterung
- Accounting und Statistik
- IP-Adressumsetzung (Masquerading, Transparent Proxying)

Einsatz Der Einsatz eines Paketfilters umfasst folgende Aspekte:

Kontrolle Mit einem Paketfilter ist es problemlos möglich, anhand von IP-Adressen nur Verbindungen zwischen bestimmten Rechnern und/oder Netzwerken zuzulassen. Damit können beispielsweise unerwünschte Zugriffe auf Webserver, die nur Werbung liefern, unterbunden werden, obwohl für diesen Fall ein Proxy wohl die bessere Lösung wäre.

Sicherheit Paketfilter bieten Schutz vor Angriffsversuchen aus fremden Netzen, indem etwa alle Verbindungsaufbauversuche von außen geblockt werden.

Wachsamkeit Manche unerwünschte Software wie Trojanische Pferde oder Spyware versuchen, vertrauliche Daten an fremde Rechner zu senden. Die Protokollierungsmechanismen eines Paketfilters erlauben es, solche Datenübertragungen aufzuspüren.

5.2 Der Paketfilter in Linux-Systemen

5.2.1 Konzeption

Geschichte Der erste Linux-Paketfilter wurde von Alan Cox Ende 1994 in die Kernelversion 1.1 implementiert und orientierte sich weitgehend an der Funktionalität des BSD-Filters `ipfw`. Seither hat der Paketfilter stetig Änderungen und Verbesserungen erfahren – in Kernel 2.0 hieß er (nach dem Verwaltungswerkzeug) `ipfwadm`, in Kernel 2.2 `ipchains` –, bis er schließlich für die Kernelversion 2.4 unter der Obhut von Paul »Rusty« Russell komplett neu geschrieben wurde. Bereits bei der Überarbeitung der Paketfilterfunktionalität für die Kernelversion 2.2 waren einige konzeptionelle Schwächen bekannt, etwa: Schwächen von `ipchains`

- Das Filterkonzept von `ipchains` ist weder modular noch erweiterbar.
- Es existiert keine bzw. lediglich eine nachträglich eingebaute, unzureichende Infrastruktur zur Weiterleitung von Paketen in den Userspace.
- Es ist mit `ipchains` nicht möglich, Filterregeln unabhängig von Schnittstellenadressen zu definieren.
- Die Einrichtung transparenter Proxies ist aufwändig und schwer zu durchschauen.
- Masquerading und sonstige Paketmanipulationen sind an die Funktionalität des Paketfilters gebunden.

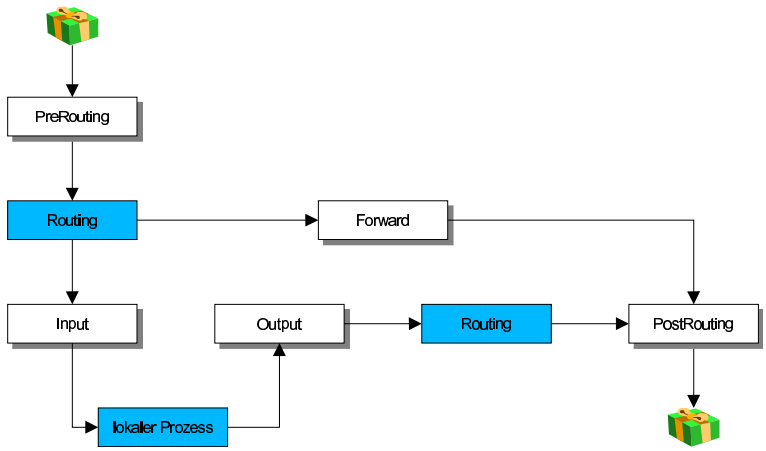


Bild 5.1: Struktur von Netfilter

Um diese Schwächen zu beseitigen, wurde für den Kernel 2.4 eine neue Infrastruktur namens »Netfilter« zur Behandlung von IP-Paketen ersonnen und programmiert. Diese hat sich als so praxistauglich erwiesen, dass für den Kernel 2.6 keine wesentlichen Veränderungen vorgenommen werden mussten [Andrews-RR03]. Netfilter

Über die Jahre sind dennoch gewisse Einschränkungen von Netfilter zutage getreten, die dazu führen, dass ein Ersatz sich am Horizont abzuzeichnen beginnt: Die »nftables«-Infrastruktur wird in Zukunft die Funktionen von Netfilter und seinen Verwandten, die Schicht-2-Filterung, ARP-Filterung und IPv6-Paketfilterung übernehmen und flexibler und effizienter gestalten.

Die grundlegende Struktur von Netfilter sieht vor, auf IP-Pakete über definierte Eingriffsmöglichkeiten (»Hooks«) an bestimmten Stellen im Protokollverbund zuzugreifen. Für IPv4 sind derzeit fünf solcher Zugriffspunkte definiert. Hooks

Bei dieser Vorgehensweise ist auch eine Priorisierung der angemeldeten Module möglich, um eine sinnvolle Reihenfolge verschiedener Operationen an einem Hook zu ermöglichen. Zur besseren Übersicht wurden die Regelwerke in drei Tabellen aufgetrennt: Priorisierung

Filter-Tabelle Diese Tabelle dient ausschließlich zur Verwaltung von Filterregeln, die bestimmen, ob ein Paket passieren darf oder nicht.

NAT-Tabelle Einträge in dieser Tabelle stellen Regeln zur *Network Address Translation*, also der Manipulation von Quell- und Zieladressen der IP-Pakete, dar.

Modifikationstabelle (engl. *mangle table*) Mit Hilfe der Regeln in dieser Tabelle lassen sich Pakete gezielt verändern, etwa um spezielle Routingverfahren umzusetzen.

Die Filter-Tabelle ist in dieser Konzeption also nur noch für die Überprüfung, nicht aber die Veränderung von IP-Paketen zuständig. Aus diesem Grund ist ein Paketfilter auf Basis von Netfilter schneller und schlanker als sein `ipchains`-Vorgänger.

Strenggenommen gibt es noch zwei andere Tabellen: Die raw-Tabelle wird vor allem für Ausnahmen von der Verbindungsverfolgung benutzt (die ansonsten einzelne Pakete existierenden Verbindungen zuzuordnen versucht), während die Security-Tabelle Zugriffsregeln zuläßt, wie sie von der spezialisierten Sicherheitsinfrastruktur SELinux verwendet werden.

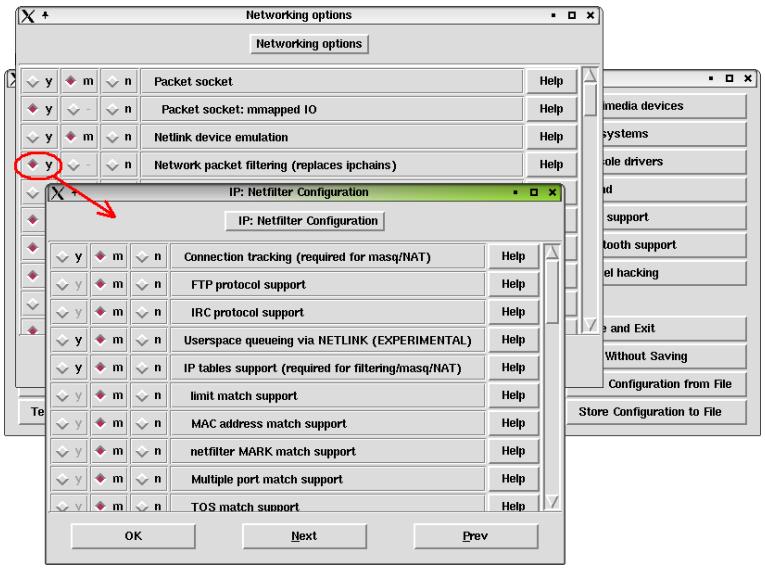


Bild 5.2: Kernelparameter für Netfilter

5.2.2 Arbeitsweise

Für die Paketfilterung, also die Entscheidung darüber, ob ein Paket weitergeleitet, an einen lokalen Prozess ausgeliefert, abgelehnt oder verworfen werden soll, ist die »Filter«-Tabelle zuständig. Wenn ein Paket auf einer Maschine ankommt, wird

Routing

der Kernel zunächst die Zieladresse ansehen und anhand dieser eine Routing-Entscheidung treffen. Die Paketfilterung wird anschließend über die drei Standard-Regelketten `INPUT`, `FORWARD` und `OUTPUT` beschrieben:

Standard-Regelketten

- Die `INPUT`-Kette erfasst Pakete, die an Prozesse auf dem Rechner selbst gerichtet sind.
- Die `OUTPUT`-Kette dient zur Kontrolle von Paketen, die von Prozessen auf dem Rechner selbst versandt werden.
- Die `FORWARD`-Kette ermöglicht die Untersuchung der Pakete, die von dem Rechner »nur« weitergeleitet werden sollen und ist daher die wohl wichtigste Kette auf einem Router.

Kette

Eine Kette ist dabei eine Ansammlung von Regeln, die festlegen, was mit einem Paket passieren soll, wenn es in den untersuchten Headers bestimmte Eigenschaften wie etwa IP-Adressen, Portnummern oder Flags aufweist. Dieses Regelwerk wird dabei eine Regel nach der anderen abgearbeitet. Passt eine Regel nicht auf ein Paket, wird die nächste abgefragt, bis eine passende Regel gefunden oder das Ende der Regelkette erreicht worden ist. In letzterem Falle greift dann die Voreinstellung der Kette, die *Policy*. Vorsichtige Gemüter werden diese immer auf `DROP` setzten und damit alles verbieten, was vorher nicht ausdrücklich erlaubt worden ist. Da im Gegensatz zu `ipchains` nun für jedes denkbare Paket exakt eine Stelle zum Filtern existiert, sind die Regelwerke ohne Einschränkung der Sicherheit wesentlich übersichtlicher als in früheren Kernels. Ferner ergibt sich aus der Möglichkeit, in der `FORWARD`-Kette sowohl die ein- als auch die ausgehende Schnittstelle angeben zu können, eine weitere Vereinfachung der Regeln.

Policy

5.2.3 Einbindung im Kernel

Bei gängigen Distributionen ist die Netfilter-Funktionalität bereits als Kernelmodule einkompiliert. Wollen Sie einen selbst kompilierten Kernel einsetzen, müssen Sie die passenden Einstellungen innerhalb der *Networking options* machen.

Einstellungen Kernelmodule

Wenn Sie die Paketfilter-Mechanismen als Kernelmodul übersetzt haben, soll-

te das Basismodul iptables_filter.ko (.o beim Kernel 2.4) nach dem ersten Aufruf von iptables automatisch geladen werden. Im Ausgangszustand sind nur die drei Standard-Filterketten INPUT, OUTPUT und FORWARD vorhanden, die keine Regeln enthalten und deren Policy auf ACCEPT steht. Sollten dennoch bereits Regeln gesetzt sein, wurden diese vermutlich durch distributionsspezifische Init-Skripte erstellt.

Übungen

5.1 [1] Stellen Sie sicher, dass Ihr Linux-Kernel Netfilter unterstützt. (Bei aktuellen Linux-Distributionen sollte das kein Thema sein.)

5.3 Das Kommandozeilenwerkzeug iptables

5.3.1 Grundlagen

Kommandozeile

Um eine möglichst einfache Verwaltung der Netfilter-Tabellen an der Kommandozeile zu ermöglichen, wurde ein einheitliches Administrationskommando namens iptables entwickelt. Dieses Kommando unterstützt Erweiterungen für neue Auswahlregeln (engl. *matches*) und Aktionen (engl. *targets*) und existiert derzeit in zwei Versionen für IPv4 und IPv6.

Die Version für IPv6 heißt – phantasievoll – ip6tables. Ihre Syntax entspricht weitestgehend der von iptables.

Der Aufbau eines iptables-Kommandos ist prinzipiell an ipchains angelehnt und umfasst mehrere, zum Teil optionale Komponenten:

- Welche Tabelle soll bearbeitet werden?
- Welche Kette in der Tabelle soll bearbeitet werden?
- Welche Operation soll an der Kette vorgenommen werden?
- Welchen Kriterien soll das Paket entsprechen (*match*)?
- Was soll mit passenden Paketen erfolgen (*target*)?

Dabei ist durch den modularen Aufbau ein große Anzahl verschiedener Optionen und Argumente möglich, die zum Beispiel im Onlinehandbuch (iptables(8)) ausführlich beschrieben sind.

Auswahl der zu bearbeitenden Tabelle Die Tabelle wird mit dem Schalter -t bzw. --table definiert. Möglich sind filter für den Paketfilter-Mechanismus, nat für die Manipulation von Quell- und Ziel-IP-Adressen bzw. -Portnummern sowie mangle für allgemeine Prä-Routing-Paketbehandlungen. Ohne Angabe der Option -t wird das Kommando auf filter, also den Paketfilter-Mechanismus, angewandt.

Auswahl der zu bearbeitenden Kette In Abhängigkeit der gewünschten Netfilter-Tabelle sind nun verschiedene Standard-Ketten verfügbar, die nicht gelöscht werden können. Dies sind im Paketfilter-Mechanismus INPUT, OUTPUT und FORWARD, wobei die Namen unbedingt in Großbuchstaben anzugeben sind. Diese Ketten enthalten im Grundzustand keine Regeln. Ferner lassen sich zusätzlich noch eigene Ketten definieren und einbinden, um etwa die Übersichtlichkeit bei komplexeren Regelwerken zu erhöhen.

Festlegung der Kettenoperation Die Kettenoperation sagt, was mit einer bestimmten Kette passieren soll und wird durch die Angabe einer Option in Großbuchstaben festgelegt. Dabei darf pro Kommandozeile üblicherweise nur eine derartige Option eingesetzt werden. Die Operationen, die sich auf eine ganze Kette beziehen, sind etwa:

Operationen für Ketten

-N, --new Eine neue Kette erstellen

-E, --rename-chain Eine Kette umbenennen

-X, --delete-chain Eine leere Kette löschen

-F, --flush Alle Regeln einer oder aller Ketten löschen

-Z, --zero Die Byte- und Paketzähler einer Kette auf Null setzen

-P, --policy Die Voreinstellung einer Kette festlegen

-L, --list Alle Regeln einer oder aller Ketten anzeigen

Operationen für einzelne Regeln

Daneben gibt es Operationen, die nur einzelne Regeln in einer Kette betreffen:

-A, --append Eine neue Regel an das Kettenende anhängen

-I, --insert Eine neue Regel in eine Kette einfügen

-R, --replace Eine Regel durch eine andere ersetzen

-D, --delete Eine Regel aus der Kette entfernen

Festlegung der Auswahlkriterien (Match) Mit Hilfe diverser Optionen lassen sich die Bedingungen festlegen, die ein Paket erfüllen muss, damit eine Regel greift. Werden in einer Kommandozeile mehrere solcher Auswahlregeln angegeben, sind diese automatisch UND-verknüpft, sie müssen also alle zutreffen, damit ein Paket erfasst wird. Um die Unterscheidung von den exklusiven Optionen zu erleichtern, wurden für diese Schalter Kleinbuchstaben gewählt. Viele Angaben lassen sich umkehren, indem ein Ausrufungszeichen als »logische Negation« vorangestellt wird.

Auswahl des Protokolls Die Option `-p` (oder `--proto`) legt das gewünschte Protokoll fest. Möglich sind die Angaben `tcp`, `udp`, `icmp` und `all` sowie (allgemein gesagt) Protokollnamen aus `/etc/protocols`, wobei hier zwischen Groß- und Kleinschreibung nicht unterschieden wird. Alternativ sind auch numerische Angaben erlaubt, aber eher unüblich. Steht ein `!` vor dem Protokoll, wird die Auswahl umgekehrt; wird `-p` nicht definiert, gilt die Vorgabe »`-p all`«. Zum Beispiele erfasst »`-p tcp`« nur TCP-Verkehr; »`-p ! udp`« alles außer UDP-Verkehr.

Auswahl von Absender und Empfänger Die Option `-s` (auch `--src` oder `--source`) legt die Absenderadresse fest. Erlaubt sind hier Netzwerknamen, Rechnernamen, Netzwerk-IP-Adressen mit Netzmaske in CIDR- oder Dotted-Quad-Schreibweise sowie Rechner-IP-Adressen. Steht ein `!` vor der Adresse, wird die Auswahl umgekehrt; wird `-s` nicht definiert, werden alle Absender erfasst, was auch mit der Angabe `0/0` möglich ist.

Die Option `-d` (`--destination` oder `--dst`) legt die Empfängeradresse fest. Die Angabe der Empfänger entspricht der Syntax der oben beschriebenen Option `-s`.

»`-d ! www.foo.bar`« erfasst zum Beispiel alle Daten, die nicht an den Rechner `www.foo.bar` gehen. »`-s 192.168.42.0/24`« erfasst alle Daten, die aus dem Netz `192.168.42.0` stammen

Auswahl der Schnittstellen `-i` (`--in-interface`) gibt die Schnittstelle an, auf der ein Paket entgegengenommen wird. Diese Option steht im Paketfilter nur in den Ketten `INPUT` und `FORWARD` zur Verfügung. Steht am Ende eines Schnittstellen-Namens ein `+`, werden alle Schnittstellen erfasst, deren Name mit der angegebenen Zeichenkette beginnt. Steht ein `!` vor dem Namen, wird die Auswahl umgekehrt; wird `-i` nicht definiert, werden alle Schnittstellen überwacht.

`-o` oder `--out-interface` gibt die Schnittstelle an, auf der ein Paket verschickt wird. Dieser Schalter ist im Paketfilter nur in den Ketten `OUTPUT` und `FORWARD` verfügbar. Die Angabe der Schnittstellen entspricht der Syntax der Option `-i`.

Zum Beispiel erfasst »`-i eth+`« alle Daten, die auf Ethernet-Interfaces eintreffen; »`-o ippp0`«erfasst alle Daten, die über die erste ISDN-Schnittstelle ausgehen.

Vielleicht hätten Sie eher »`eth*`« statt »`eth+`« erwartet; die +-Syntax erlaubt jedoch eine leichtere Regeleingabe über die Kommandozeile, da die Shell »`eth*`« als Dateinamen expandieren würde.

Fragmentierung Das IP-Protokoll bietet die Möglichkeit, große Pakete in kleinere Bruchstücke (Fragmente) zu zerteilen, die erst auf dem Zielrechner wieder zusammengefügt werden. Da nur im ersten Bruchstück vollständige Paketinfomationen (IP- und TCP/UDP/ICMP-Header) vorliegen – die späteren Fragmente enthalten nur einen IP-Header –, kann keine Regel, die beispielsweise Portangaben enthält, auf Folgefragmente angewendet werden. Die Option `-f` (oder `--fragment`) erfasst nicht das erste, sondern alle weiteren Bruchstücke fragmentierter IP-Pakete. Steht ein `!` vor der Option, gilt die Regel hingegen für das erste Bruchstück oder unfragmentierte Pakete.

Wenn Sie *connection tracking* verwenden (siehe unten), setzt die Netfilter-Infrastruktur die Fragmente zusammen, bevor Ihre Regeln angeschaut werden. In diesem Fall sind keine `-f`-Regeln notwendig.

Übungen

5.2 [2] `iptables` erlaubt die Angabe einer Aktion (Regel hinzufügen, löschen, Kette hinzufügen, löschen, …) pro Aufruf. Welche Gründe könnten für diesen Ansatz sprechen und gegen die Idee, dass `iptables` beim Start eine Konfigurationsdatei mit Regelspezifikationen liest, so wie viele andere Programme das tun?

5.3.2 Erweiterungen

Das modulare Konzept von Netfilter ermöglicht es, die vorhandenen Funktionen problemlos um neue zu ergänzen. Solche Erweiterungen bestehen aus zwei Teilen, nämlich einem Kernelmodul und einer Schnittstelle für die Kommandozeile.

Modulaufruf

Bereits im Standardumfang stehen eine Reihe von Modulen bereit, welche die Paketauswahl betreffen. Diese rufen Sie durch die Angabe der Option `-m` bzw. `--match` explizit mit dem Modulnamen auf; protokollspezifische Erweiterungen werden schon durch die `-p`-Option implizit geladen. Je nach Modul sind so eine Reihe weiterer Auswahloptionen möglich. Eine Übersicht dieser neuen Optionen erhalten Sie, wenn Sie nach dem Modulaufruf der Schalter `-h` bzw. `--help` eingeben.

Da sich die Liste der verfügbaren Module permanent ändert, sind hier nur die wichtigsten Vertreter in alphabetischer Reihenfolge aufgeführt. Für weitere Informationen sollten Sie zunächst die Handbuchseiten Ihrer Distribution zu Rate ziehen. Eine weitere nützliche Informationsquelle sind die Web-Seiten des Netfilter-Projekts [Netfilter-Ext-HOWTO].

icmp Das ICMP-Modul wird durch »`-p icmp`« geladen. Als neue Auswahlregel erlaubt dann `--icmp-type` die Angabe des gewünschten ICMP-Protokolltyps bzw. -codes als numerischer Wert oder in ausgeschriebener Form. Verwenden Sie etwa

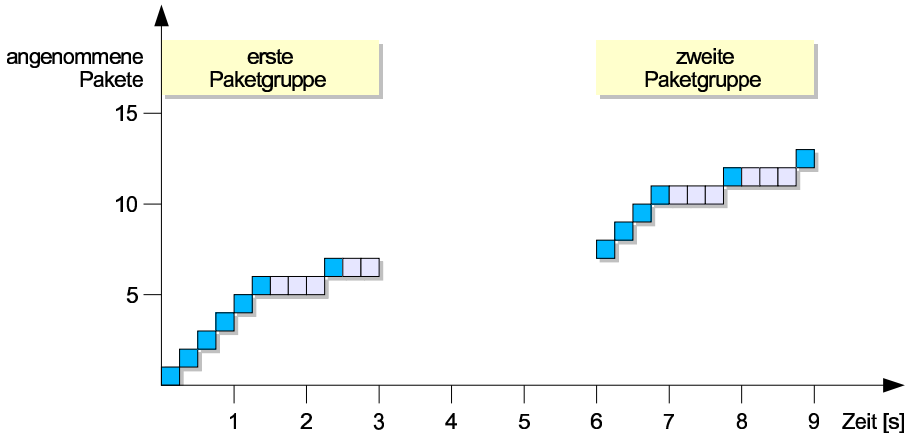


Bild 5.3: Beispiel für das `limit`-Modul

»`-p icmp --icmp-type pong`«, um alle Antworten auf Ping-Anfragen (`echo-reply`) zu erfassen; »`-p icmp --icmp-type 3/1`« erfasst alle "*host unreachable*"-Meldungen.

length Mit Hilfe dieses Moduls kann die Paketgröße überwacht werden. Nach Laden des Moduls mit »`-m length`« kann mit `--length` die Paketgröße in Byte als bestimmter Wert oder Bereichsangabe in der Form ⟨*von*⟩:⟨*bis*⟩ festgelegt werden. Nützlich ist dies etwa, um übergroße Ping-Pakete abzufangen: »`-m length --length 512`« erfasst zum Beispiel nur Pakete mit einer Mindestgröße von 512 Bytes.

zeitliche Begrenzung

limit Das `limit`-Modul erlaubt eine zeitliche Begrenzung der von einer Regel erfassten Pakete. Dies kann nicht nur gegen Denial-of-Service-Angriffe oder Portscans verwendet werden, sondern schützt auch vor ausufernden Protokolldateien.

Nach Laden des Moduls mit »`-m limit`« existieren zwei neue Optionen. Mit `--limit` kann die durchschnittliche Anzahl von Ereignissen pro Zeiteinheit festgelegt werden, Vorgabewert ist hier drei pro Stunde (`3/hour`). Zeitangaben sind ebenso in den Einheiten `second`, `minute` oder `day` möglich. Der Maximalwert für die Erfassung kann nach `--limit-burst` festgelegt werden, Standardwert ist 5.

Die Standardwerte hätten zur Folge, dass die ersten 5 Pakete von einer Regel mit Standardlimit erfasst werden, danach wird nur alle 20 Minuten (3 pro Stunde) je ein Paket erfasst. Um das Limit wieder komplett zu löschen, darf 100 Minuten lang kein passendes Paket ankommen. Dies ergibt sich, wenn man den Wert für `--limit-burst` durch den für `--limit` teilt, hier also $5/(3/h) = 100$ min.

Noch ein Beispiel: Betrachten Sie die Optionen »`-m limit --limit 1/s`«. Die Durchschnittsrate liegt also hier bei einem Paket pro Sekunde, der Maximalwert unverändert bei 5 Paketen. Nun kommen drei Sekunden lang je 4 Pakete pro Sekunde an, danach herrscht für drei Sekunden Ruhe, schließlich treffen erneut für drei Sekunden je 4 Pakete pro Sekunde ein (Bild 5.3).

Zunächst werden also 6 Pakete angenommen (5 gemäß des Maximalwerts und eines aus der Durchschnittsrate `1/s`), danach nur immer ein Paket pro Sekunde. Drei Sekunden Pause genügen, damit das System drei Pakete »vergisst«. Darum werden in der nächsten dreisekündigen Sendephase zunächst vier Pakete (die drei »vergessenen« sowie wieder eines gemäß der Durchschnittsrate `1/s`) und dann wieder je ein Paket pro Sekunde angenommen.

multiport Dieses Modul erlaubt die Angabe von bis zu 15 Quell- oder Zielports und kann nur in Verbindung mit den Optionen »`-p tcp`« bzw. »`-p udp`« eingesetzt werden. Nach »`-m multiport`« stehen dann die Optionen `--source-ports` bzw. `--sports` für Quellports, `--destination-ports` bzw. `--dports` für Zielports oder einfach `--ports` für identische Quell- und Zielports bereit. Die Ports werden dabei als kommase-

parierte Liste aufgeführt. Zum Beispiel: »-p tcp -m multiport --dports 80,443« erfasst HTTP- und HTTPS-Pakete zu Webservern.

state Dieses sehr nützliche Modul ist ursprünglich für die NAT-Funktionalität entwickelt worden, kann aber auch in Filterregeln eingesetzt werden. Über dieses *Connection-Tracking*-Modul werden alle Verbindungen mit Quell- und Zieladressen bzw. -ports in einer Tabelle protokolliert. Mit Hilfe des state-Moduls kann so der Zustand von Verbindungen in Filterregeln abgefragt werden. Nach Aktivierung mit »-m state« stehen für --state folgende Verbindungszustände zur Wahl:

NEW Das Paket initialisiert eine neue Verbindung.

ESTABLISHED Das Paket gehört zu einer bereits aufgebauten Verbindung.

RELATED Das Paket steht in Beziehung zu einer bereits aufgebauten Verbindung, ist aber nicht direkt Teil davon. Zu dieser Kategorie gehören etwa ICMP-Fehlermeldungen oder FTP-Datenverbindungen – letzteres Beispiel erfordert allerdings ein spezielles Modul ip_conntrack_ftp.

INVALID Das Paket kann keinem der vorher aufgeführten Zustände zugeordnet werden.

»-m state --state ESTABLISHED,RELATED« zum Beispiel erfasst alle Pakete, die zu bereits etablierten Verbindungen gehören.

tcp Das TCP-Modul wird nach Angabe von »-p tcp« automatisch geladen und erlaubt folgende Analysen:

--source-port, --sport Festlegung des Absenderports als Portname gemäß /etc/services, als numerischer Wert oder als Bereichsangabe in der Form ⟨*von*⟩:⟨*bis*⟩. Fehlt bei letzterem eine Angabe, wird als Untergrenze 0, als Obergrenze 65535 angenommen. Mit einem ! kann die Auswahl negiert werden. Pro Regel ist nur eine Quellport-Angabe erlaubt, sollen mehrere, nicht zusammenhängende Ports in einer Regel erfasst werden, muss die multiport-Erweiterung verwendet werden.

--destination-port, --dport Festlegung des Empfängerports analog zu --sport.

--tcp-flags Ermöglicht die Analyse der Flags im TCP-Header und erwartet zwei Argumente. Zunächst sind die Flags anzugeben, die überwacht werden sollen. Das zweite Argument sagt dann, welche der überwachten Flags gesetzt sein müssen.

--syn Dient zur Erfassung von TCP-Verbindungsaufbaupaketen und entspricht in seiner Funktion dem Schalter »--tcp-flags SYN,ACK,RST SYN«. Wieder kann mit einem vorangestellten ! die Auswahl umgekehrt werden.

--tcp-option Erlaubt die Analyse anhand bestimmter TCP-Optionen, die als Nummer angegeben werden müssen [IANA-TCP-Param].

Einige Beispiele: »-p tcp --sport 1024: --dport 80« erfasst alle TCP-Pakete, die von einem hoch numerierten Port (1024–65535) an einen Webserver (Zielport 80) gerichtet sind. »-p tcp --tcp-flags ACK,FIN FIN« erfasst FIN-Scans von Portscannern wie nmap. »-p tcp ! --syn« erfasst keine TCP-Verbindungsaufbaupakete.

udp Nach Aufruf durch »-p udp« bietet das UDP-Modul zwei Auswahlkriterien, die analog zu denen des TCP-Moduls funktionieren:

--source-port, --sport Festlegung des Absenderports als Portname gemäß /etc/services, als numerischer Wert oder als Bereichsangabe in der Form ⟨*von*⟩:⟨*bis*⟩. Fehlt bei letzterem eine Angabe, wird als Untergrenze 0, als Obergrenze 65535 angenommen. Mit einem ! kann die Auswahl negiert werden.

Pro Regel ist nur eine Quellport-Angabe erlaubt, sollen mehrere, nicht zusammenhängende Ports in einer Regel erfasst werden, muss die `multiport`-Erweiterung verwendet werden.

`--destination-port, --dport` Festlegung des Empfängerports analog zu `--sport`.

Zum Beispiel erfasst »`-p udp --dport 53`« an DNS-Server gerichtete UDP-Pakete (Zielport 53).

Übungen

5.3 [1] Geben Sie `iptables`-Optionen an, die auf die folgenden Pakete passen:

1. Alle ICMP-`ECHO-REQUEST`-Pakete (vulgo »Ping«)
2. Eingehende Pakete, die versuchen, eine SSH-Verbindung aufzubauen
3. Ausgehende Pakete an HTTP-Server im Adressbereich `10.0.0.0`

5.4 [2] Was ist der Vorteil des `state`-Moduls im Vergleich zu individuellen Regeln?

5.3.3 Festlegung der Aktion

Nachdem ein Paket anhand seiner Headereigenschaften ausgewählt worden ist, muss natürlich noch definiert werden, was mit diesem Paket passieren soll. Dies wird im Englischen als "*target*" bezeichnet und nach der Option `-j` bzw. `--jump` angegeben, es erfolgt also ein Sprung zu einer Aktion oder in eine benutzerdefinierte Kette.

Eingebaute Aktionen

Fest eingebaut stehen folgende Aktionen bereit:

`ACCEPT` Dient dazu, ein Paket passieren zu lassen.

`DROP` Verwirft ein Paket, ohne dem Absender eine Rückmeldung zu geben.

`QUEUE` leitet das Paket an ein Programm außerhalb des Kernels weiter, falls der Kernel dies unterstützt.

`RETURN` springt das Ende einer Kette an. In einer benutzerdefinierten Kette erfolgt daraufhin der Rücksprung zur ursprünglichen Kette, während in den drei Standardketten die Voreinstellung (*policy*) greift.

Benutzerdefinierte Ketten

Grundsätzlich lassen sich alle Regeln in den drei Standardketten unterbringen. Netfilter ermöglicht aber auch die Verwendung von benutzerdefinierten Ketten, die einige Annehmlichkeiten bieten:

Übersichtlichkeit Je mehr Regeln definiert werden, desto unübersichtlicher wird das Regelwerk. Eine Verteilung der Regeln auf benutzerdefinierte Ketten erhöht die Überschaubarkeit durch Verringerung der Regelanzahl in den Standardketten.

Vereinfachung Sollen etwa identische Regeln für mehrere unterschiedliche Netze vergeben werden, ist es sinnvoll, eine eigene Kette mit den für die verschiedenen Netze identischen Parametern zu definieren, statt diese Parameter für jedes Netz wiederholt angeben zu müssen.

Leistung Da die Standardketten immer von oben nach unten nach passenden Regeln durchforstet werden, nimmt bei zunehmender Kettenlänge natürlich auch die durchschnittliche Suchzeit nach einer passenden Eintragung zu. Diese kann durch Verteilung der Regeln auf benutzerdefinierte Ketten verringert werden.

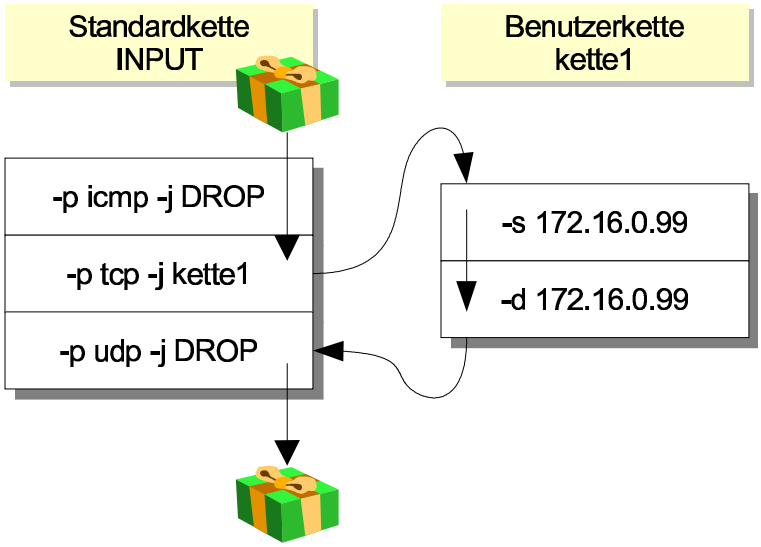


Bild 5.4: Benutzerketten in Netfilter

Einrichten

Benutzerdefinierte Ketten müssen zunächst eingerichtet werden, bevor dort Regeln untergebracht werden können. Dies gelingt mit der Option `-N` bzw. `--new-chain`. Der Name einer solchen Kette sollte in Kleinbuchstaben geschrieben werden, um Kollisionen mit den eingebauten Ketten zu vermeiden, und darf natürlich noch nicht belegt sein. Ferner darf seine Länge 31 Zeichen nicht überschreiten.

Löschen

Nicht mehr benötigte benutzerdefinierte Ketten können mit der Option `-X` bzw. `--delete-chain` wieder entfernt werden. Dies gelingt allerdings nur, wenn die Ketten keine Regeln mehr enthalten und keine Verweise auf diese Ketten in anderen Ketten enthalten sind.

Beispiel

In Bild 5.4 empfängt eine Maschine ein an sie gerichtetes TCP-Paket, das vom Absender `172.16.0.99` stammen möge. Zuständig für die Annahme ist die `INPUT`-Kette. Die erste Regel dort greift nicht, wohl aber die zweite. Dort wird als Aktion die benutzerdefinierte Kette `kette1` aufgeführt. Dort passt zwar die erste Regel (Absender war ja `172.16.0.99`), aber da keine Aktion angegeben ist, durchläuft das Paket dennoch die nächste Regel. Diese greift nicht, also ist das Kettenende erreicht, wodurch in die ursprüngliche Kette zurückgesprungen wird. Die dritte Regel dort passt ebenfalls nicht, folglich entscheidet nun die Voreinstellung der `INPUT`-Kette über das weitere Schicksal des Pakets.

Erweiterungen

Ebenso wie bei den Auswahlregeln unterstützt Netfilter auch für die Aktionen modulare Erweiterungen. Dies sind für den Paketfiltermechanismus unter anderem folgende:

Protokollierung

`LOG` Diese Aktion ermöglicht die Protokollierung von Paketen über das Kernel-Log (und damit normalerweise den `syslogd`). Im Gegensatz zu den üblichen Aktionen wird die Abarbeitung der Regelkette nicht beendet, wenn eine passende `LOG`-Regel gefunden wird. Um also ein Paket zu protokollieren und zu verwerfen, sind zwei Regeln mit identischen Auswahlregeln nötig, wovon die erste dann zur Protokollierung, die zweite zum Verwerfen dient. `LOG` erlaubt ferner noch einige Optionen, die wichtigsten sind:

- **`--log-level`** Festlegung der Prioritätsstufe als numerische Angabe oder ausgeschrieben (siehe `syslog.conf`(5)).
- **`--log-prefix`** Festlegung einer bis zu 29 Zeichen langen Erläuterung zur besseren Analyse der Protokollmeldungen.

Als Beispiel: »`-j LOG --log-prefix "Paketfilter"`« protokolliert alle Pakete, ergänzt um die Erläuterung »Paketfilter«.

Fehlermeldung

`REJECT` Im Gegensatz zu `DROP` verwirft `REJECT` ein Paket nicht stillschweigend, sondern schickt dem Absender eine ICMP-Fehlermeldung. Diese kann mit der

Option `--reject-with` genauer definiert werden, wobei neben diversen ICMP-Meldungen auch die Möglichkeit besteht, ein TCP-Reset zurückzusenden. Ohne Angabe der Option `--reject-with` erhält der Absender die ICMP-Meldung "*port unreachable*" (Typ 3, Code 3). Mit »`-j DROP --reject-with icmp-host-unreachable`« zum Beispiel werden erfasste Pakete verworfen, und der Absender erhält eine ICMP-Fehlermeldung vom Typ 3, Code 1.

Übungen

5.5 [!2] Wie würden Sie dafür sorgen, dass mit `DROP` verworfene Pakete immer erst protokolliert werden?

5.6 [2] Unter gewissen Umständen schickt die Aktion `REJECT` keine ICMP-Fehlermeldung zurück, sondern verwirft ein Paket einfach. Wann passiert das und warum?

5.7 [3] Untersuchen Sie den Unterschied von `DROP` und `REJECT`. Richten Sie dazu auf einen Rechner die Filterregeln

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 81 -j DROP
iptables -A INPUT -p tcp --dport 82 -j REJECT --reject-with tcp-reset
```

ein. Verbinden Sie sich nun per `netcat` (oder `telnet`) der Reihe nach mit den Ports 80, 81 und 82 und achten Sie auf Unterschiede.

Benutzen Sie zusätzlich den Portscanner `nmap` und scannen Sie die drei Ports.

5.8 [3] Im Paketfilter-Skript der SUSE-Distribution wird statt einem einfachen

```
iptables … -j DROP
```

die folgende `REJECT`-Sequenz benutzt:

```
iptables … -p tcp -j REJECT --reject-with tcp-reset
iptables … -p udp -j REJECT --reject-with icmp-port-unreachable
iptables … -j REJECT --reject-with icmp-proto-unreachable
```

Was ist der Vorteil gegenüber einem einfach `DROP`?

5.3.4 Operationen auf eine komplette Kette

Neben der Möglichkeit, benutzerdefinierte Ketten zu erstellen bzw. wieder zu entfernen, werden in der Praxis einige andere Operationen häufiger durchgeführt werden, die sich auf komplette oder gar alle Ketten auswirken.

Voreinstellung

Die Voreinstellung der Ketten wird mit der Option `-P` beeinflusst. Nur die drei Standardketten erlauben die Festsetzung einer Policy, benutzerdefinierte Ketten springen bei Erreichen des Kettenendes in die aufrufende Kette zurück. Mögliche Policy-Einstellungen sind `ACCEPT` oder `DROP`. Mit

```
# iptables -P FORWARD DROP
```

zum Beispiel werden alle Pakete verworfen, für die keine passende Weiterleitungsregel vorliegt.

Anzeige

Die Anzeige aller derzeit vorhandenen Regeln ist mit der Option `-L` möglich. Dabei sorgt `-v` bzw. `--verbose` für eine ausführlichere Ansicht, die auch ein- und ausgehende Schnittstellen sowie Paket- und Bytezähler darstellt. Um gerundete Werte exakt auszugeben, ist ferner die Angabe von `-x` bzw. `--exact` möglich.

Schließlich sorgt -n bzw. --numeric dafür, dass Netzwerk- und Rechneradressen und Portnummern numerisch ausgegeben werden.

Sollen die Zählerstände zurückgesetzt werden, kann dies mit der Option -Z erfolgen. Zählerstände

Zum Beispiel:

```
# iptables -L INPUT          Voreinstellung und Regeln der INPUT-Kette
# iptables -L -n -v          Voreinstellung und Regeln aller Ketten, numerisch
```

Mit Hilfe des Schalters -F lassen sich alle Regeln einer Kette oder gar alle Regeln in einem Schritt entfernen. Entfernen

5.3.5 Sichern der Filterregeln

Nach der Eingabe einer Filterregel wird diese lediglich im Arbeitsspeicher abgelegt (vergleichbar z. B. mit den Routing-Informationen). Dies hat zur Folge, dass nach einem Neustart des Sytems keine Filterregeln mehr vorhanden sind. Um sich nun die mühsame manuelle Eingabe aller Regeln zu ersparen, sind zwei unterschiedliche Lösungsansätze möglich.

Einerseits können Sie ein Init-Skript verwenden, in dem die gesamten iptables-Befehle einzeln aufgeführt sind. Diese Variante hat den Vorteil, dass nicht nur die Befehle leicht korrigiert werden können, sondern auch eine einfache Übertragung des Skriptes auf andere Maschinen möglich ist. Ein solches Init-Skript wird im Praxisbeispiel (Abschnitt 5.3.6) beschrieben. Init-Skript

Alternativ können Sie die derzeit gesetzten Regeln mit Hilfe des Kommandos iptables-save inklusive Policies und Zählerständen auf der Standardausgabe ausgeben. Leiten Sie diese Informationen in eine Datei um, können Sie deren Inhalt mit iptables-restore wieder einlesen. Auch dieses Verfahren läßt sich mit einem Init-Skript automatisieren, ist in der Praxis jedoch selten anzutreffen. Hauptursache dafür mag die geringere Flexibilität dieser Variante sein. iptables-save

Eine wichtige Beobachtung ist, dass Sie Filterregeln definieren dürfen, deren Kriterien auf nicht vorhandene Netzwerkschnittstellen Bezug nehmen. Das macht es möglich, die Filterregeln in Kraft zu setzen, *bevor* Sie das Netz initialisieren. Dadurch gibt es kein »Verwundbarkeitsfenster«, während dem das Netz läuft, aber noch keine Filterregeln gelten.

5.3.6 Praxisbeispiel

Zur Veranschaulichung der Umsetzung von Paketfilterregeln mit Hilfe eines init-Skriptes mag folgendes Beispiel dienen. Ein Netzwerk, etwa 192.168.0.0/24, sei über eine Maschine an das Internet angebunden. Diese hat nach innen die IP-Adresse 192.168.0.254, nach außen hingegen 1.2.3.4. Auf diesem Rechner soll kein Routing erfolgen, sondern lediglich den Clients der Zugriff auf Webserver im Internet durch einen HTTP-Proxy, etwa squid, erlaubt werden. Vom internen Netz her darf die Maschine mit ping von allen Clients auf Erreichbarkeit getestet werden, ferner soll von der Maschine 192.168.0.99 ausgehend ein Fernzugriff per ssh möglich sein. Wie könnte nun ein Init-Skript aussehen, welches genau diese Zugriffe gestattet, alle anderen Pakete hingegen verwirft?

Wir definieren zunächst einige Variable zur (späteren) Vereinfachung:

```
#!/bin/sh
# /etc/init.d/netfilter

# Variablen definieren
#
# lokales Netz
LOC_NET="192.168.0.0/24"
# externes Netz
```

```
EXT_NET="0/0"
# lokales Interface
LOC_IF="eth0"
# externes Interface
EXT_IF="eth1"
# lokale IP-Adresse
LOC_IP="192.168.0.254"
# externe IP-Adresse
EXT_IP="1.2.3.4"
# vertrauenswürdige interne Maschine
ADMINHOST="192.168.0.99"
# externer DNS-Server des Providers
DNSSERVER="2.3.4.5"
```

Als Init-Skript sollte unser Skript so organisiert sein, dass es mit dem Parameter `start` aufgerufen den Paketfilter aktiviert, mit `stop` dagegen ihn ausschaltet.

Aktivierung

Betrachten wir zunächst den Aktivierungsfall:

```
case "$1" in
start)
  echo -n "Starting netfilter rules "

  # Policies setzen
  iptables -t filter -P INPUT DROP
  iptables -t filter -P OUTPUT DROP
  iptables -t filter -P FORWARD DROP

  # Schutz vor Synflooding aktivieren
  echo 1 > /proc/sys/net/ipv4/tcp_syncookies
  # Reaktion auf Broadcast-Pings deaktivieren
  echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
  # Reaktion auf seltsame ICMP-Pakete deaktivieren
  echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
```

Policies

Es ist sinnvoll, als allererstes die Policies zu etablieren, insbesondere `DROP`. Die Kernel-Parameter helfen gegen verschiedene Arten von netzbasierten Angriffen.

SYN-Flooding

SYN-Flooding ist eine (inzwischen etwas antiquierte) Form von Denial-of-Service-Angriff: Die meisten TCP/IP-Implementierungen sehen einen endlichen Puffer für die Speicherung von Daten über »halboffene« Verbindungen vor – solche, für die ein `SYN`-Paket empfangen und mit `SYN/ACK` quittiert wurde, für die aber das dritte Paket des Drei-Wege-Handshake noch nicht eingegangen ist. Ist dieser Puffer erschöpft, können keine neuen Verbindungen angenommen werden (der Rechner ist vom Netz aus nicht mehr erreichbar), aber TCP/IP schreibt vor, dass eine bestimmte Zeit (70 Sekunden) gewartet werden muss, bevor die Verbindung verloren gegeben werden darf. Der Angriff besteht darin, einem Rechner eine große Anzahl von `SYN`-Paketen mit einer gefälschten, nicht benutzten IP-Absenderadresse zu schicken, damit der Puffer aufgefüllt wird. Der Rechner muss dann den TCP-Timeout abwarten und ist in der Zwischenzeit nicht erreichbar. Als Angriff ist SYN-Flooding heute eigentlich zu plump, dient aber oft dazu, einen Rechner temporär unerreichbar zu machen, um dessen IP-Adresse für andere Zwecke zu missbrauchen.

syn cookies

Als mögliche Abhilfe kann Linux im Falle eines SYN-Flooding-Angriffs sogenannte *syn cookies* einsetzen, bei denen lokal nichts gespeichert werden muss, sondern die notwendigsten Informationen kryptographisch in die initiale Folgenummer codiert werden, die der angesprochene Rechner dem Client zurückgibt. Ist das beantwortete Paket Teil des Angriffs, so kommt

darauf nie eine Antwort, aber das macht nichts; ist das beantwortete Paket Teil einer ernstgemeinten Verbindung, kann aufgrund der Informationen im vom Client geschickten *dritten* Paket des TCP-Drei-Wege-Handshake, insbesondere der bestätigten Folgenummer des Servers, sichergestellt werden, dass es sich um eine echte Antwort auf das zweite Paket handelt, und die Verbindung angenommen werden. Als Technik sind *syn cookies* nicht unumstritten, da sie der reinen Exegese der einschlägigen RFCs zuwider laufen, aber sie sind so wirkungsvoll, dass pragmatisch gegen ihren Einsatz nichts Gravierendes spricht.

ICMP-Broadcasts

Auf ICMP-Broadcasts sollten Sie nicht unbedingt reagieren; diese sind gerne Teil eines »Smurf«-Angriffs[1], bei dem ein Angreifer von einer gefälschten Adresse aus Mengen von ICMP-Broadcast-Ping-Paketen schickt. Die angepingten Rechner antworten an die gefälschte IP-Adresse und überfluten den betreffenden Rechner mit Ping-Echos.

ICMP-Antworten auf Broadcast-Pakete

Die Zuweisung an `icmp_ignore_bogus_error_responses` führt dazu, dass der Rechner ICMP-Antworten auf Broadcast-Pakete ignoriert, die von manchen fehlerhaft implementierten Routern ausgehen. Üblicherweise würden diese Pakete im Systemlog protokolliert werden; diese Einstellung dient dazu, Plattenplatz zu sparen.

Als nächstes setzen wir einige schnittstellenspezifische Parameter:

```
# Schutz vor ICMP-Redirects, Source-Routing und IP-Spoofing;
# protokolliere merkwürdige Pakete
for d in /proc/sys/net/ipv4/conf/*
do
        echo 0 >$d/accept_redirects
        echo 0 >$d/accept_source_route
        echo 1 >$d/rp_filter
        echo 1 >$d/log_martians
done
```

Redirection

Über `accept_redirects` wird gesteuert, ob der Rechner ICMP-Nachrichten beachtet, die ihm bessere Routen zu bestimmten anderen Netzen oder Rechnern vorschlagen (sollte er nicht); mit `accept_source_route` bestimmen wir, dass *source routing* nicht akzeptiert wird (gute Idee); `rp_filter` prüft im wesentlichen, dass die Absenderadressen von Paketen zur aktuellen Routingtabelle passen; in unserem Beispiel würden zum Beispiel Pakete verworfen, die mit einer Absender-IP-Adresse im `192.168.0.0/24`-Netz auf der »externen« Netzwerkkarte eingehen. Schließlich sorgt `log_martians` dafür, dass alle Pakete mit »unmöglichen« Adressen protokolliert werden. Eine »unmögliche« Adresse ist eine Adresse, die aufgrund der aktuellen Routingtabelle nicht geroutet werden kann.

IP-Spoofing

»Unmögliche« Adressen

Lokale Kommunikation

Prozesse auf dem lokalen Rechner sollten auch über TCP/IP miteinander kommunizieren können. Aus diesem Grund erlauben wir den beliebigen Austausch von Paketen über die `lo`-Schnittstelle:

```
# loopback freischalten
iptables -t filter -A INPUT -s 127.0.0.0/8 -d 127.0.0.0/8 \
    -i lo -j ACCEPT
iptables -t filter -A OUTPUT -s 127.0.0.0/8 -d 127.0.0.0/8 \
    -o lo -j ACCEPT
```

Bei der Formulierung von Regeln im allgemeinen hilft uns das *connection tracking* mit dem `state`-Modul. Wir erklären einfach Pakete, die Teil einer existierenden Verbindung sind (`ESTABLISHED`) oder anderweitig zu ihr gehören (`RELATED`) für

[1] Die wörtliche deutsche Übersetzung »Schlumpf« hat sich im Jargon nicht durchgesetzt.

in Ordnung. Später müssen wir uns dann nur noch um das Aufbauen von Verbindungen (Status NEW) kümmern. Dies halbiert im wesentlichen die Anzahl der erforderlichen Regeln:

```
# Antworten generell zulassen
iptables -t filter -A INPUT -m state --state ESTABLISHED,RELATED \
    -j ACCEPT
iptables -t filter -A OUTPUT -m state --state ESTABLISHED,RELATED \
    -j ACCEPT
```

Ping

Jetzt kommen wir dazu, tatsächlich Funktionalität kontrolliert einzuschalten. Als erstes erlauben wir den lokalen Benutzern, den Paketfilter-Rechner anzupingen:

```
# ping von innen auf Firewall erlauben
iptables -t filter -A INPUT -s $LOC_NET -d $LOC_IP \
    -p icmp --icmp-type ping -i $LOC_IF -j ACCEPT
```

SSH-Zugriffe

Als nächstes gestatten wir die SSH-Zugriffe vom Rechner des Administrators aus:

```
# ssh von innen durch $ADMINHOST auf Firewall erlauben
iptables -t filter -A INPUT -s $ADMINHOST -d $LOC_IP \
    -p tcp --sport 1024: --dport 22 -i $LOC_IF \
    -m state --state NEW -j ACCEPT
```

Web-Proxy

Rechner im lokalen Netz müssen sich an den lokalen Web-Proxy wenden können. Dieser braucht dann entsprechend Zugang ins Internet, um im Namen der lokalen Benutzer Seiten holen zu dürfen:

```
# HTTP von innen nach auf squid erlauben
iptables -t filter -A INPUT -s $LOC_NET -d $LOC_IP \
    -p tcp --sport 1024: --dport 3128 -i $LOC_IF \
    -m state --state NEW -j ACCEPT

# HTTP von squid nach außen erlauben
iptables -t filter -A OUTPUT -s $EXT_IP -d $EXT_NET \
    -p tcp --sport 1024: --dport 80 -o $EXT_IF \
    -m state --state NEW -j ACCEPT
```

DNS-Zugriff

DNS-Zugriff braucht im einfachsten Fall auch nur der Web-Proxy. (Die Rechner im internen Netz haben wohl eine /etc/hosts-Datei.)

```
# DNS-Anfragen zum $DNSSERVER erlauben
iptables -t filter -A OUTPUT -s $EXT_IP -d $DNSSERVER \
    -p udp --sport 1024: --dport 53 -o $EXT_IF \
    -m state --state NEW -j ACCEPT
iptables -t filter -A OUTPUT -s $EXT_IP -d $DNSSERVER \
    -p tcp --sport 1024: --dport 53 -o $EXT_IF \
    -m state --state NEW -j ACCEPT
```

(Beachten Sie, dass wir sowohl UDP als auch TCP für DNS frei schalten. Normalerweise verwendet DNS UDP, fällt aber für große Antworten, typischerweise ab 512 Bytes, auf TCP zurück.)

Protokoll

Zum Schluss protokollieren wir alle Pakete, die noch übrigbleiben (sie werden als nächstes von der Policy der Ketten, DROP, entsorgt).

```
# alle zu routenden Pakete zum Debuggen protokollieren
iptables -A FORWARD -j LOG
```

```
# alles erledigt...
echo "...done"
;;
```

Wenn unser Init-Skript mit dem Parameter `stop` aufgerufen wird, sollte es den Ursprungszustand des Systems wieder herstellen: Deaktivierung

```
stop)
    echo -n "Shutting down netfilter rules "

    # alle Regeln leeren
    iptables -t filter -F

    # alle Benutzerketten entfernen
    # iptables -t filter -X

    # Schutz vor Synflooding deaktivieren
    echo 0 > /proc/sys/net/ipv4/tcp_syncookies
    # Reaktion auf Broadcast-Pings aktivieren
    echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
    # Reaktion auf seltsame ICMP-Pakete aktivieren
    echo 0 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

    # Schutz vor ICMP-Redirects, Source-Routing, IP-Spoofing abschalten
    for d in /proc/sys/net/ipv4/conf/*
    do
        echo 1 >$d/accept_redirects
        echo 1 >$d/accept_source_route
        echo 0 >$d/rp_filter
        echo 0 >$d/log_martians
    done

    # Policies auf ACCEPT setzen
    iptables -t filter -P INPUT ACCEPT
    iptables -t filter -P OUTPUT ACCEPT
    iptables -t filter -P FORWARD ACCEPT

    # alles erledigt...
    echo "...done"
    ;;
```

Hier sind zum Schluss noch ein paar typische weitere Aktionen für Init-Skripte: weitere Aktionen

```
# Regelwerk zurücksetzen und neu einrichten
  restart)
      $0 stop
      $0 start

      # alles erledigt...
      echo "...done"
      ;;
  status)
      echo "Checking for netfilter rules"

      # Zustand des Paketfilters ausführlich und numerisch anzeigen
      iptables -t filter -L -v -n
      ;;
  *)
```

```
      echo "Usage: $0 start|stop|status|restart"
      exit 1
      ;;
esac
```

Variable

Wie Sie sehen, werden in diesem Skript innerhalb der Regeln Variable für bestimmte Adressen bzw. Interfaces benutzt. Dadaurch lassen sich zwar die einzelnen Regelzeilen etwas schwieriger nachvollziehen, es steigt jedoch die Flexibilität des Skriptes. Sollte es etwa erforderlich werden, den internen IP-Adressenbereich zu ändern, lässt sich dies an einer zentralen Stelle im Skript vornehmen, ohne jede Regel einzeln anfassen zu müssen.

5.4 Adressumsetzung (Network Address Translation)

5.4.1 Anwendungsfälle für NAT

Wie beschrieben sorgt ein Router dafür, dass Pakete, deren Ziel nicht im lokalen Netzwerksegment liegt, an den jeweiligen Empfänger weitergereicht werden. Dabei bleiben die Quell- und Zieladressen im IP-Header unangetastet, der Router dekrementiert lediglich die Paketlebensdauer (»TTL«) und berechnet eine neue Prüfsumme.

Manipulationen an den IP-Adressen

In manchen Fällen ist es jedoch erforderlich, auch Manipulationen an den IP-Adressen durchzuführen. Denkbare Szenarien wären etwa:

Verbergen der internen Netzwerkadressen nach außen Dieses Vorgehen ist zwingend erforderlich, wenn im lokalen Netz private IP-Adressbereiche eingesetzt werden. Ein gängiger Anwendungsfall ist etwa der Anschluss eines Netzwerkes an das Internet über eine Einwahlverbindung (Modem, ISDN, DSL). Aber auch bei der Verwendung von offiziellen IP-Adressen kann ein solches Vorgehen sinnvoll sein, um die Struktur des internen Netzwerks nicht nach außen preiszugeben.

Verknappung der offiziellen IP-Adressen Offizielle IP-Adressen stellen heutzutage ein für Firmen und Organisationen nur begrenzt verfügbares Gut dar. Reichen nun die vorhandenen IP-Adressen nicht aus, um alle von außen zugänglichen Server ansprechen zu können, kann diese Problematik durch Umschreiben der Zieladressen entschärft werden.

Transparent Proxying Um zu erreichen, dass alle Rechner über auf einen Dienst über einen Proxy zugreifen, können Zieladressen und ggf. auch Portnummern umgeschrieben werden. So lässt sich etwa der komplette HTTP-Verkehr über einen `squid`-Proxy abwickeln, ohne dass dieser im Browser des Clients eingetragen werden muss. Die Bezeichnung »Transparent Proxying« weist darauf hin, dass der Client auf diese Weise einen Proxy verwendet, ohne dies zu bemerken oder gar umgehen zu können.

5.4.2 Varianten von NAT

Prinzipiell lassen sich bei der Adressumsetzung zwei Fälle unterscheiden:

Absenderadresse ändern

Source NAT Durch Umschreibung der Absenderadresse können Funktionen wie das Verbergen der internen Netzwerkstruktur oder Masquerading durchgeführt werden. Solcherlei Anpassungen werden in der Kette `POSTROUTING` vorgenommen, also erst kurz bevor ein Paket den Rechner verlässt. Ein auf derselben Maschine laufender Paketfilter sieht daher stets die ursprüngliche Absenderadresse.

Zieladresse ändern

Destination NAT Änderungen der Zieladresse für das Umleiten auf andere Maschinen oder Ports (etwa für Transparent Proxying) finden für über das Netzwerk empfangene Pakete in der Kette `PREROUTING` statt, also sogleich,

nachdem ein Paket auf dem Rechner eingetroffen ist. Läuft auf der selben Maschine ein Paketfilter, sieht dieser also die bereits umgeschriebene Zieladresse. Auch für lokal generierte Pakete ist das Ändern von Zieladresse bzw. -port möglich, wobei dies dann in der OUTPUT-Kette erfolgen muss.

Eine Übersicht hierzu kann Bild 5.1 entnommen werden.

5.4.3 NAT per Netfilter

Für die Umschreibung von IP-Adressen und/oder Portnummern ist im Linux-System die Netfilter-Suite zuständig. Zur Manipulation der NAT-Regeln dient wie gewohnt das Kommando iptables, wobei die Angabe der NAT-Tabelle mit »-t nat« hier zwingend erforderlich ist. Für die Umschreibung muss jeweils nur das erste Paket einer neuen Verbindung das Regelwerk durchlaufen. Findet sich dabei eine passende Regel, werden alle weiteren Pakete der Verbindung automatisch analog behandelt, ohne das Regelwerk erneut bemühen zu müssen. Dies ergibt eine deutliche Verbesserung der Systemleistung, erfordert jedoch eine Funktionalität, die es dem Kernel erlaubt, den Verbindungszustand festzustellen. Das zu diesem Zweck entwickelte conntrack-Modul führt also gewissermaßen Buch über die derzeit etablierten Verbindungen und ermöglicht auch die Handhabung verbindungsloser Protokolle wie UDP oder ICMP. Da diese Fähigkeit für Paketfilter ebenfalls hochinteressant ist, kann dieses Netfilter-Modul auch dort, wie beschrieben, mit »-m state« eingesetzt werden.

NAT-Tabelle

conntrack-Modul

Umschreiben des Absenders Änderungen des Absenders (SNAT) werden in der POSTROUTING-Kette definiert, indem die Aktion SNAT angegeben wird. Damit stehen mit Hilfe der Option --to-source Möglichkeiten zur Umschreibung der IP-Adresse und/oder der Ports bereit. Eine Zeile wie

POSTROUTING-Kette

Aktion SNAT

```
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 1.2.3.4
```

würde dafür sorgen, dass alle Pakete, die den Rechner über das Interface eth0 verlassen, als Absender die IP-Adresse 1.2.3.4 bekommen.

Neben der Angabe einer einzelnen IP-Adresse kann auch ein ganzer Adressbereich verwendet werden, etwa

Adressbereich

```
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT \
>    --to-source 1.2.3.4-1.2.3.8
```

In diesem Fall werden die verfügbaren IP-Adressen von 1.2.3.4 bis 1.2.3.8 abwechselnd verwendet.

Ferner kann auch ein gewisser Bereich an Ports für die Umschreibung definiert werden, zum Beispiel

Portbereich

```
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT \
>    --to-source 1.2.3.9:1-1023
```

Ein sehr gängiger Sonderfall von SNAT ist Masquerading, also das Verbergen aller interner Adressen hinter einer einzigen, dynamisch zugewiesenen, offiziellen Adresse. Hierfür existiert eine besondere Aktion. Um etwa bei einer ISDN-Einwahlverbindung alle ausgehenden Pakete mit der bei der Einwahl vom Provider zugewiesenen Adresse zu versehen, genügt ein einfaches Kommando wie

Masquerading

```
# iptables -t nat -A POSTROUTING -o ippp0 -j MASQUERADE
```

MASQUERADE versieht alle Pakete mit der aktuellen Adresse des Ausgangs-Interfaces. Dies hat den zusätzlichen Vorteil, dass diese Adresse nicht hart kodiert werden muss, was ärgerlich wäre, da sie bei Einwahlzugängen ja in der Regel ständig wechselt.

Demaskierung

Übrigens ist es in keinem Fall erforderlich, Antwortpakete durch irgendwelche NAT-Befehle wieder an die ursprünglichen Absender zu leiten, diese »Demaskierung« übernimmt der Kernel automatisch.

PREROUTING-Kette

Umschreiben des Empfängers Änderungen des Ziels (DNAT) werden für empfangene Pakete in der PREROUTING-Kette vorgenommen. Die Umschreibung des Emfängers von lokal generierten Paketen hingegen erfolgt in der OUTPUT-Kette.

Aktion DNAT

In beiden Fällen ist die Aktion DNAT anzugeben. Somit stehen mit der Option --to-destination Funktionen zur Anpassung von Zieladresse und/oder -port zur Verfügung. Dabei ist wie beim Umschreiben der Absenderadresse die Angabe eines einzelnen Empfängers oder die Angabe eines Adressbereiches möglich,

Lastverteilung

die abwechselnd verwendet werden. So kann eine einfache Lastverteilung auf mehrere Zielmaschinen erreicht werden:

```
# iptables -t nat -A PREROUTING -i eth1 -j DNAT \
>    --to-destination 1.2.3.4-1.2.3.8
```

Transparenter Proxy

Um einen transparenten Proxy für den kompletten HTTP-Verkehr zu konfigurieren, könnte eine Zeile ähnlich dieser verwendet werden:

```
# iptables -t nat -A PREROUTING -i eth0 \
>    -p tcp --dport 80 -j DNAT --to-destination 10.0.0.100:8080
```

Jeglicher TCP-Verkehr, der an externe Server auf deren Port 80 gerichtet ist, wird also umgeleitet auf die Maschine 10.0.0.100, auf deren Port 8080 ein transparenter Proxy seinen Dienst verrichtet.

Sollte das Ziel des umgeleiteten Verkehrs nicht auf einer anderen Maschine liegen, sondern einen Dienst auf einem bestimmten Port der lokalen Maschine darstellen, ist dies wiederum ein Sonderfall. Für diesen Zweck existiert eine eigene

REDIRECT

Aktion namens REDIRECT, die etwa für einen transparenten Proxy folgendermaßen eingesetzt werden kann:

```
# iptables -t nat -A PREROUTING -i eth0 \
>   -p tcp --dport 80 -j REDIRECT --to-port 3128
```

Hier wird der gesamte TCP-Verkehr zu Zielport 80 auf einen lokalen Squid umdirigiert, der auf seinem Standardport 3128 lauscht.

5.4.4 Besonderheiten von NAT

Zum besseren Verständnis beschreiben wir hier noch einige grundlegende Verhaltensweisen der NAT-Funktionalität:

- Bei der Angabe einer Reihe von IP-Adressen zur Umschreibung wird vom Kernel diejenige ausgewählt, für die momentan die wenigsten registrierten Verbindungen vorliegen. Auf diese Weise wird eine primitive Lastverteilung erreicht.

Aktion ACCEPT

- Um zu verhindern, dass eine bestimmte Verbindung von NAT-Prozessen erfasst wird, kann einfach eine passende Regel mit der Aktion ACCEPT definiert werden.

- Üblicherweise wird versucht, innerhalb der vorgegebenen Regeln so wenig wie möglich am Paket-Header zu ändern. Das bedeutet etwa, dass Portnummern nur dann umgeschrieben werden, wenn dies unumgänglich ist.

Konflikte

- Auch bei Verbindungen, die normalerweise nicht von NAT-Regeln erfasst werden, kann der Kernel bei Konflikten Umschreibungen vornehmen. Dies wäre etwa in diesem Szenario der Fall:

1. Ein interner Rechner `192.168.0.99` baut von seinem Port 1024 eine HTTP-Verbindung zu `www.kernel.org`, Port 80 auf.
2. Der Masquerading-Router schreibt die Absenderadresse auf seine externe IP-Adresse `1.2.3.4` um, den Quellport 1024 lässt er unverändert.
3. Der Masquerading-Router `1.2.3.4` möchte von seinem Port 1024 selbst eine HTTP-Verbindung zu `www.kernel.org`, Port 80 aufbauen.
4. Die NAT-Funktionalität des Kernels ändert daraufhin den Quellport der zweiten Verbindung auf 1025 ab, um einen Konflikt zu vermeiden.

Für diesen Fall sind die Portnummern in drei Gruppen aufgeteilt, nämlich in Ports unterhalb von 512, Ports von 512 bis 1023 sowie Ports ab 1024. Ein Umschreiben der Portnummern wird nur innerhalb dieser Gruppen erfolgen, d. h., es wird niemals Port 666 auf Port 4711 verändert werden.

- Der Kernel ist mächtig genug, um Adresskonflikte nach Möglichkeit zu verhindern. Es ist also erlaubt, zwei oder mehr private IP-Adressen in eine offizielle zu übersetzen, was bei Masquerading ja auch erfolgt. Ferner kann ein privates, internes Netz in einen offiziellen Adressbereich umgeschrieben werden, obwohl dieser bereits ebenfalls von internen Clients verwendet wird, etwa:

```
# iptables -t nat -A POSTROUTING -s 192.168.42.0/24 \
>    -o eth1 -j SNAT --to-source 200.200.200.0/24
```

- Falls das Ändern der Adressen und/oder Portnummern nicht ohne Konflikte möglich ist, werden die jeweiligen Pakete vom Kernel verworfen. Dies geschieht ebenfalls, sofern Pakete nicht eindeutig einer Verbindung zugeordnet werden können, da sie beispielsweise beschädigt sind oder der NAT-Router nicht mehr über ausreichend Arbeitsspeicher verfügt.

- Der Einsatz von NAT bedingt, dass der komplette ein- und ausgehende Verkehr über den NAT-Router abgewickelt wird. Andernfalls kann eine ordnungsgemäße Funktion nicht gewährleistet werden.

- Bei Änderungen der Absenderadresse per SNAT muss dafür gesorgt werden, dass alle Antwortpakete wieder den SNAT-Router erreichen können. Dies ist auf verschiedene Arten möglich:

  – Wenn für SNAT die eigene Adresse des NAT-Routers verwendet wird und für diese bereits alle benötigten Routen etc. gesetzt wurden, sind keine weiteren Maßnahmen erforderlich.

  – Wird für SNAT eine unbenutzte Adresse in lokalen LAN verwendet (etwa `1.2.3.222` im Netz `1.2.3.0/24`), ist es erforderlich, dass der NAT-Router auf ARP-Anfragen zu dieser Adresse reagiert. Dies kann einfach durch Zuweisung eines IP-Alias für die entsprechende Schnittstelle erfolgen, etwa mit:

    ```
    # ifconfig eth0:0 1.2.3.222 up
    ```

  – Dient eine völlig andere Adresse für SNAT, ist es erforderlich, dass alle Zielmaschinen entsprechende Routen zum SNAT-Router besitzen. Dies ist dann der Fall, wenn es sich bei dem NAT-Router um deren Standard-Gateway handelt, ansonsten sind den Zielmaschinen manuell oder per Routing-Protokoll die entsprechenden Informationen beizubringen.

- Liegt das Ziel von per DNAT manipulierten Paketen im gleichen Netz wie der Absender, wird der Empfänger die Antworten nicht über den NAT-Router, sondern direkt an den Client zurücksenden, die dieser dann natürlich verwirft. Klassisches Beispiel für diesen Fall wäre der Zugriff eines

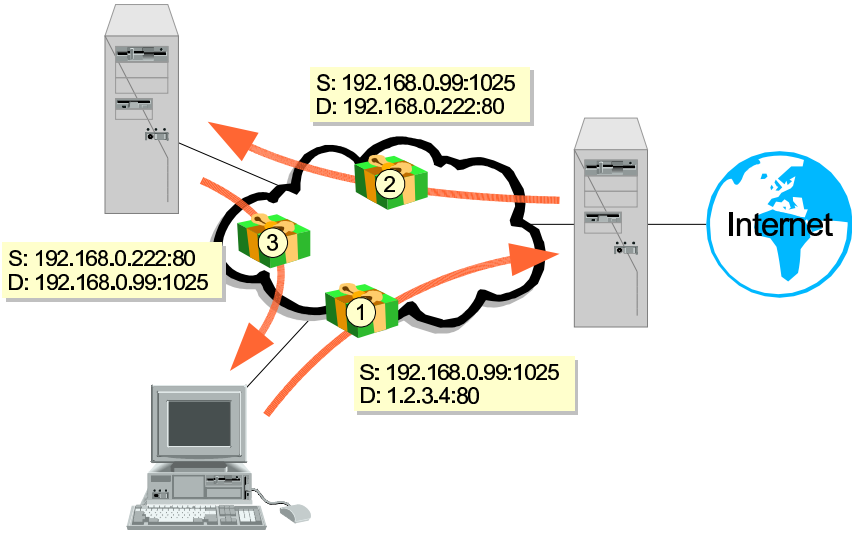


Bild 5.5: Destination NAT

internen Rechners, hier `192.168.0.99`, auf den unternehmenseigenen »externen« Webserver unter `1.2.3.4`, der in Wirklichkeit allerdings die Maschine `192.168.0.222` im internen Netz darstellt.

Auf dem NAT-Router sorgt dabei eine Regel wie diese für das Umschreiben des Empfängers:

```
# iptables -t nat -A PREROUTING -d 1.2.3.4 -p tcp \
>    --dport 80 -j DNAT --to-destination 192.168.0.222
```

Um dieser Problematik Herr zu werden, sind zwei unterschiedliche Lösungsansätze denkbar:

- Zum einen wäre es möglich, einen internen DNS-Server so zu konfigurieren, dass Anfragen von innen bezüglich des »externen« Webservers gleich die interne IP-Adresse, hier `192.168.0.222`, zurückliefern. Die Clients greifen dann ohne den Umweg über den NAT-Router direkt auf den richtigen Zielrechner zu.
- Die andere Möglichkeit besteht darin, den NAT-Router zusätzlich noch den Absender auf seine eigene interne IP-Adresse umschreiben zu lassen. Somit werden die Antworten nicht direkt an die Clients, sondern über den Umweg über den NAT-Router zugestellt. Unter der Annahme, der NAT-Router habe die interne IP-Adresse `192.168.0.254`, könnte eine passende Netfilter-Regel folgendermaßen lauten:

  ```
  # iptables -t nat -A POSTROUTING \
  >    -s 192.168.0.0/24 -d 192.168.0.222 -p tcp --dport 80 \
  >    -j SNAT --to-source 192.168.0.254
  ```

  Da `POSTROUTING`-Regeln erst ganz am Ende des Regelwerks abgearbeitet werden, fällt bei genauer Betrachtung des Beispiels auf, dass hier bereits die in der `PREROUTING`-Kette geänderte Zieladresse, jedoch die noch unveränderten Absenderadressen herangezogen werden müssen.

- Für NAT bei besonderen Protokollen sind je zwei Netfilter-Erweiterungen erforderlich. Eine dient dabei für die Erfassung des Verbindungszustands, die andere zur Änderung der Header. Die für FTP erforderlichen Module `ip_conntrack_ftp.o` und `ip_nat_ftp.o` gehören bereits zum Standardumfang von Netfilter und erlauben die Verwendung von aktivem sowie passivem FTP.

Übungen

5.9 [!3] Ein Netzwerk, etwa `192.168.0.0/24`, sei über eine Maschine per ISDN-Einwahlverbindung an das Internet angebunden. Dieser Router hat nach innen die IP-Adresse `192.168.0.254`, nach außen hingegen eine dynamische IP-Adresse, die vom Provider bei der Einwahl vergeben wird. Die internen Rechner müssen hinter der externen IP-Adresse verborgen werden. – Den Clients im internen Netz soll der Zugriff auf beliebige Webserver im Internet erlaubt werden. Vom internen Netz her darf der Router selbst mit `ping` von allen Clients angesprochen werden. Die internen Rechner sollen auch Rechner im Internet per `ping` erreichen können. Auf dem Router selbst läuft ein Caching-Only-DNS-Server, der Anfragen der internen Rechner an den DNS-Server `2.3.4.5` des Providers weiterreicht. Ferner soll von der internen Maschine `192.168.0.99` ausgehend ein Fernzugriff per ssh möglich sein. Wie könnte nun ein Init-Skript aussehen, welches genau diese Zugriffe gestattet, alle anderen Pakete hingegen verwirft?

Kommandos in diesem Kapitel

`iptables-restore` Setzt eine gespeicherte Netfilter-Konfiguration in Kraft `iptables-restore`(8) 77

`iptables-save` Sichert die aktuelle Netfilter-Konfiguration `iptables-save`(8) 77

Zusammenfassung

- Paketfilter dienen zur Kontrolle von Verbindungen zwischen bestimmten Rechnern und Netzen, zum Schutz vor Angriffsversuchen und zur Überprüfung von Netzzugriffen durch lokal installierte Software.
- Linux unterstützt schon lange Paketfilterfunktionalität. Die aktuelle Implementierung, Netfilter, ist leistungsfähig, flexibel und erweiterbar und entspricht dem *state of the art*.
- Netfilter erlaubt unter anderem Paketfilterung, Adressumsetzung sowie das fast beliebige Umschreiben von Paketdaten.
- Mit Netfilter werden bestimmte Eingriffspunkte in der Verarbeitung von Datagrammen definiert, an denen Regelketten abgearbeitet werden können. Diese Regeln bestehen jeweils aus einem Auswahl- und einem Aktionsteil; wenn der Auswahlteil auf das betrachtete Datagramm passt, wird die Aktion ausgeführt.
- Das Kommando `iptables` dient zur Konfiguration von Netfilter.
- Adressumsetzung erlaubt unter anderem das Verbergen interner Netzadressen (»Masquerading«), die Implementierung transparenter Proxies oder Lastverteilung über mehrere äquivalente Server.

Literaturverzeichnis

Andrews-RR03 Jeremy Andrews. »Interview: Rusty Russell«, September 2003. http://kerneltrap.org/node/892

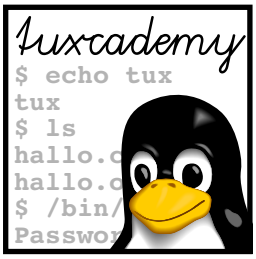
Bar03 Wolfgang Barth. *Das Firewall Buch*. SuSE Press, 2003, 2. Auflage. ISBN 3-899900-44-8. http://www.suse.de/de/private/products/books/3_899900_44_8/

IANA-TCP-Param »TCP Option Numbers«. http://www.iana.org/assignments/tcp-parameters

NAT-HOWTO Rusty Russell. »Linux 2.4 NAT HOWTO«, Januar 2002. http://www.netfilter.org/documentation/HOWTO/de/NAT-HOWTO.html

Netfilter-Ext-HOWTO Fabrice Marie. »Netfilter Extensions HOWTO«. http://www.netfilter.org/documentation/HOWTO/netfilter-extensions-HOWTO.html

Packet-Filtering-HOWTO Rusty Russell. »Linux 2.4 Packet Filtering HOWTO«, Mai 2000. http://www.netfilter.org/documentation/HOWTO/de/packet-filtering-HOWTO.html



6

Sicherheitsanalyse

Inhalt

Lernziele

- Zweck und Probleme von Sicherheitsanalysewerkzeugen einschätzen können
- Den Portscanner nmap einsetzen können
- Den Sicherheitsscanner OpenVAS kennen und betreiben können

Vorkenntnisse

- Linux-Administrationskenntnisse
- TCP/IP- und Netzadministrationskenntnisse

6.1 Einleitung

Werkzeuge, die zum Untersuchen eines Netzwerkes benutzt werden, existieren schon seit einiger Zeit. Als Beispiel sei hier nur das *Security Analysis Tool for Auditing Networks* von Wietse Venema und Dan Farmer, kurz »SATAN« genannt, das schon seit 1995 im Internet frei verfügbar ist (siehe `http://www.porcupine.org/satan/`). Alternativen bzw. Nachfolger dieses Programmes, das potentielle Sicherheitslücken aufdecken kann, werden zum Teil als frei erhältliche Software (SARA, `nmap`, OpenVAS), aber auch als kommerzielle Produkte (SAINT, Nessus) angeboten. Ob es sich bei diesen Programmen um Angriffs- oder Verteidigungswerkzeuge handelt, bleibt der jeweiligen Betrachtungsweise überlassen. Sowohl Cracker als auch Systemverwalter können solche Software nutzen, um sich Informationen über die aktiven Rechner und Dienste in einem Netz zu verschaffen und eventuelle Schwachstellen zu finden.

SATAN

Mit allen genannten Programmen ist es möglich, so genannte Portscans durchzuführen. Hierbei wird überprüft, welche Dienste auf welchen Rechnern zur Verfügung stehen, indem dort einfach eine ganze Reihe von Ports willkürlich angesprochen und die Antworten des gescannten Rechners ausgewertet werden. Weiter gehende Tests sollen dann zeigen, ob die Dienste bestimmte, bereits bekannt gewordene Schwächen aufweisen, die potentielle Angreifer für ihre Zwecke missbrauchen können. Ein spezielles Programm, das solche Schwächen ausnutzt, oder auch der Vorgang dieses Ausnutzens heißt *exploit*.

Sowohl die kommerziellen Produkte als auch die frei verfügbare Software bieten die Möglichkeit, sich Berichte über die gefundenen Sicherheitslücken anfertigen zu lassen. Aufgrund dieser Berichte können Sie dann entscheiden, welche Dienste durch neuere Versionen ersetzt oder welche ganz abgeschaltet werden sollten.

»Tiger Teams«, »Pen-Tester« oder andere Sicherheitsberater bieten das Scannen von Rechnern als Dienstleistung an (sogenannte *security audits*), mit der etwa Firewall-Konfigurationen überprüft werden können. Es versteht sich von selbst, dass die Vergabe entsprechender Aufträge nach außen ebenfalls wieder ein Sicherheitsrisiko darstellt – *quis custodiet ipsos custodes?*

Übungen

6.1 [!3] »Sicherheitsscanner sind reine Crackerwerkzeuge und sollten nicht frei für jeden auf dem Internet zur Verfügung stehen.« Diskutieren Sie diese Aussage.

6.2 Netzanalyse mit `nmap`

6.2.1 Grundlagen

Der Network Mapper `nmap` ist ein Werkzeug, mit dem einzelne Rechner, aber auch komplette Netze gescannt werden können. Dabei werden besondere IP-Pakete verwendet, um verschiedene Daten zu sammeln. Neben einem Kommandozeilenwerkzeug existiert auch ein grafisches Frontend, beide Programmteile sind quelloffene Software unter der GPL. Die jeweils aktuellste Programmversion für Linux, diverse Unixe und Windows sowie ausführliche Dokumentationen finden sich im WWW unter `http://www.insecure.org/nmap/`. Wichtige Leistungsmerkmale des Programmes sind etwa:

Dienst-Erkennung Wie es sich für einen ordentlichen Portscanner gehört, klappert `nmap` bei einem Durchlauf verschiedene Ports eines Rechners ab und registriert die Reaktion des gescannten Hosts auf die Anfragen. Mit Hilfe einer Datenbank, die derzeit einige tausend Einträge umfasst, werden dann die üblicherweise auf

diesem Port arbeitenden Dienste ausgegeben. In seltenen Fällen stimmt diese Zuordnung nicht, da einige Administratoren gerne Dienste auf ungewöhnliche Ports umbiegen, um Angreifern das Auffinden zu erschweren. Abhilfe bietet die Versionserkennung durch nmap.

Versionserkennung Nicht nur offene Ports, sondern auch die Version des zugehörigen Programms werden ermittelt. Zu diesem Zweck verwendet nmap wiederum eine beiligende Datenbank. Die Versionserkennung gelingt nicht nur bei klassischen UDP- bzw. TCP-Diensten, sondern sogar bei mit SSL verschlüsselten Varianten, falls nmap auf OpenSSL zugreifen kann. Ferner können Sie auch RPC-basierende Dienste untersuchen. Dies erlaubt die Analyse von Diensten, die auf ungewöhnlichen Portnummern laufen und hilft einem Angreifer, die richtigen *exploits* auszuwählen.

Betriebssystemerkennung Die *operating system detection* ermöglicht mit recht hoher Verlässlichkeit das Bestimmen des Betriebssystems auf den untersuchten Rechnern. Dabei wird per *TCP/IP fingerprinting* das unterschiedliche Verhalten der Betriebssysteme auf verschiedene Scanverfahren mit einer internen Datenbank abgeglichen und danach ein entsprechender Vorschlag ausgegeben.

Zu den in nmap eingebauten Analysemethoden gehören unter anderem die Analyse von TCP-Folgenummern, verfügbaren TCP-Optionen, die anfängliche TCP-Fenstergröße sowie die Reaktion auf FIN-Pakete. Eine ausführliche Übersicht aus der Feder von Fyodor, dem Autor von nmap, ist auf der Web-Seite von nmap zugänglich [Fyo98].

Verschiedene Scan-Varianten Während normalen Benutzern lediglich der TCP-Connect-Scan zur Verfügung steht, welcher letztendlich einem automatisierten telnet auf eine Reihe verschiedener Ports gleichzusetzen ist, darf der Systemadministrator root zwischen vielen Scan-Varianten auswählen. So genannte *stealth scans* führen keinen kompletten TCP-Drei-Wege-Handshake durch, sondern senden im dritten Schritt des Verbindungsaufbaus manipulierte Pakete. Dies führt dazu, dass der Zugriff nicht durch den Dienst selbst protokolliert wird und so der Scan eher vor den Augen eines Administrators verborgen bleibt. Daneben können auch UDP- oder RPC-Scans durchgeführt werden.

Auch das zeitliche Verhalten von nmap kann gesteuert werden. Im Normalfall wird nmap den Zielrechner so schnell wie möglich scannen. Liegt das Hauptaugenmerk aber darauf, eine Analyse möglichst unauffällig durchzuführen, kann die Wartezeit bis zur Abfrage des nächsten Ports stufenweise verlängert werden. Naturgemäß steigt dadurch die Dauer des Gesamtvorgangs deutlich an, die einzelnen Zugriffe auf die verschiedenen Ports verschwinden aber gewissermaßen im "Grundrauschen"des Netzwerks und sind damit kaum noch als Scan zu enttarnen.

Lockvögel und lebende Tote Die Analyse eines Hosts bedingt, dass die gescannte Maschine Antwortpakete an den Initiator des Scans senden muss. Ein aufmerksamer Administrator oder ein Angriffserkennungssystem werden daher in der Lage sein, den Urheber des Scans zu ermitteln und ggf. per Firewall auszusperren. Um dies zu erschweren, lassen sich in nmap mit der Option -D *decoys*, also Lockvögel, angeben. Für den gescannten Host sieht es dann so aus, als ob er von einer ganzen Reihe Rechner gleichzeitig analysiert wird, der eigentliche Initiator ist nicht herauszulesen.

Mittels eines Idle-Scans per Option -sI ist es unter bestimmten Umständen (im Wesentlichen vorhersagbare IP-Identifikationsnummern) sogar möglich, einen Scan komplett über einen Mittler, den so genannten *zombie host*, abwickeln. Dieser Rechner scheint dabei der Ausgangspunkt der Analyse zu sein, während der eigentliche Urheber nicht mit der Zielmaschine kommuniziert und so völlig verborgen bleibt. Eine ausführliche Beschreibung des Vorgangs und der Randbedingungen ist wiederum auf den nmap-Seiten nachzulesen [Fyo02].

6.2.2 Syntax und Optionen

allgemeine Syntax

Die allgemeine Syntax der Kommandozeilenvariante lautet

```
nmap [⟨Optionen⟩] ⟨Ziel⟩ [⟨Ziel⟩ …]
```

Zur Angabe eines oder mehrere Ziele stehen diverse Varianten bereit. Neben Rechnernamen oder IP-Adressen einzelner Rechner können wie gewohnt auch Netze mit Netzmasken in Bitschreibweise, etwa `172.16.0.0/16`, verwendet werden. Ferner lassen sich auch IP-Adressbereiche angeben, wobei das Sonderzeichen `*` für alle möglichen Werte steht. Die Angabe `172.16.*.*` entspräche also den oben genannten privaten Klasse-B-Netz, während eine Angabe wie etwa `172.16.0-22,23,24-255.*` zwar komplizierter aussieht, aber den gleichen Bereich überprüfen würde.

Sie können diverse Optionen auf der Kommandozeile angeben. Diese werden in der Dokumentation ausführlich vorgestellt. Hier greifen wir daher nur die wichtigsten Optionen heraus.

Scantypen

Zunächst die verschiedenen Scantypen:

`-sT` Diese Variante ist als einzige auch für normale Benutzer verfügbar und führt einen gewöhnlichen TCP-Verbindungsaufbau durch. Ein solches »automatisiertes telnet« auf die Zielports wird üblicherweise in verschiedenen Protokolldateien auftauchen und kann daher leicht entdeckt werden.

`-sS` Beim halboffenen Scanning wird der dritte Schritt des Drei-Wege-Handshakes durch ein Paket ersetzt, das das RST-Flag gesetzt hat. Es wird daher keine TCP-Verbindung etabliert und dementsprechend auch nicht protokolliert. Dieses *stealth scanning* ist die Vorgabe für privilegierte Benutzer.

`-sF` Da viele Firewalls Verbindungsaufbaupakete von außerhalb blocken, wird beim `FIN`-Scan ein unaufgefordertes TCP-Paket mit gesetzten `FIN`-Flag gesendet. Geschlossene Ports sollten darauf mit einem `RST` reagieren, während offene Ports laut [RFC0793] solche Pakete ignorieren müssten.

`-sX` Der Christmas-Tree-Scan ähnelt sehr der Variante `-sF`, hat aber im TCP-Paket sozusagen den gesamten Weihnachtsbaum erleuchtet, also die Flags `FIN`, `URG` und `PSH` gesetzt.

`-sN` Der Nullscan ist wiederum eine Spielart von `-sF`, diesmal aber mit einem Paket, in welchem kein einziger TCP-Flag gesetzt ist.

`-sP` Der Ping-Scan testet lediglich mit verschiedenen Verfahren die Erreichbarkeit der Zielmaschine, führt aber keinen wirklichen Scan durch.

`-sU` Statt TCP-Ports lassen sich so UDP-Ports untersuchen. Geschlossene Ports antworten auf die UDP-Pakete mit der ICMP-Fehlermeldung »Port unreachable«; erhält `nmap` eine reguläre oder gar keine Antwort, wird der Port als offen oder geblockt angesehen.

Ein UDP-Scan gestaltet sich oftmals recht langwierig, da viele Rechner eine Begrenzung der ICMP-Fehlermeldungen pro Zeiteinheit, ein *rate limit*, eingebaut haben.

`-sR` Bei einem RPC-Scan werden die gefundenen TCP- bzw. UDP-Ports mit RPC-NULL-Kommandopaketen beehrt. Handelt es sich tatsächlich um RPC-Ports, gibt `nmap` Programm und Versionsnummer ähnlich »`rpcinfo -p`« aus. Im Falle eines Versionsscans wird »`nmap -sR`« automatisch durchgeführt.

`-sV` Die Versionserkennung wird versuchen, die hinter den ermittelten offenen Ports steckenden Dienste herauszufinden. Zu diesem Zweck greift `nmap` auf eine Datei zurück, in der die verschiedenen Untersuchungsmethoden definiert sind. Nach Möglichkeit werden Protokoll, Programmname, Versionsnummer und ggf. weitere Informationen gesammelt und ausgegeben. Dies funktioniert selbst mit Diensten, die per SSL verschlüsselt kommunizieren.

Eine Erreichbarkeitsprüfung realisieren die folgenden Optionen: Erreichbarkeitsprüfung

-P0 Im Regelfall wird nmap vor dem Scan die Erreichbarkeit des Opfers prüfen. Ist dies nicht gewünscht, etwa weil etwa eine dazwischen liegende Firewall Ping-Pakete blockiert, lässt sich dieser Test mit der Option -P0 unterbinden.

-PE nmap sendet ICMP-Echo-Requests an den Zielhost, verwendet also das klassische Ping zur Erreichbarkeitsprüfung.

-PA Statt klassischer ICMP-Pings werden TCP-ACK-Pakete an Port 80 geschickt, um die Erreichbarkeit des Zielrechners zu testen. Falls gewünscht, sind auch andere Zielports für diese Prüfung möglich.

-PS Vergleichbar mit -PA verwendet nmap hier zur Analyse TCP-Verbindungsaufbaupakete.

-PB Die Voreinstellung von nmap entspricht der Kombination der Schalter -PA und -PE.

Die folgenden Optionen steuern die Scangeschwindigkeit: Scangeschwindigkeit

-T Mit sechs sehr blumigen Schlüsselworten oder nüchternen Zahlen können Sie nach dem Schalter -T das zeitliche Verhalten eines Scandurchlaufs festlegen. Es stehen zur Auswahl:

- **0, paranoid** Die Ports werden einer nach dem anderen ausprobiert, wobei zwischen den Paketen jeweils eine mindestens fünfminütige Sendepause eingelegt wird.
- **1, sneaky** Diese Einstellung ähnelt Paranoid, die Wartezeit zwischen dem Versand von Paketen beträgt aber nur 15 Sekunden.
- **2, polite** Wiederum werden die Ports einer nach dem anderen geprüft, allerdings nur mit einer Sendepause von jeweils mindestens 0,4 Sekunden. Dies ist ungefähr eine Größenordnung langsamer als die Standardgeschwindigkeit.
- **3, normal** Per Voreinstellung versucht nmap, die Analyse so schnell wie möglich vorzunehmen, ohne dabei das Netzwerk zu überlasten.
- **4, aggressive** Bei bestimmten Scanverfahren, vor allem SYN-Scans auf Rechner, die viele Ports filtern, können ungeduldige Naturen mit dieser Einstellung eine Menge Zeit sparen.
- **5, insane** Der schnellste Modus wartet niemals länger als 0,3 Sekunden auf eine Antwort und ist daher nur in schnellen Netzen sinnvoll, andernfalls leidet die Verlässlichkeit der Analyse.

Hier noch ein paar weitere Optionen: weitere Optionen

-O Mittels *TCP/IP fingerprinting* versucht nmap, den Betriebssystemtyp des Zielrechners zu ermitteln. Verschiedene Experimente werden dabei durchgeführt und die Ergebnisse mit einer Datenbank abgeglichen. Findet sich dort kein passender Eintrag, bietet nmap an, die Resultate an die Entwickler zu senden, damit die Datenbank entsprechend ergänzt werden kann.

Neben dem Betriebssystem der Zielmaschine ermittelt nmap noch deren Betriebsdauer (per TCP-Timestamp-Option) und wagt eine Aussage über die Wahrscheinlichkeit, die TCP-Folgenummern des gescannten Hosts vorhersagen zu können. Je einfacher der Zusammenhang zwischen zwei aufeinanderfolgenden Folgenummern ist, desto leichter fällt es einem Angreifer, einen Folgenummernangriff zu starten. Die Bewertungen gehen dabei von "*trivial joke*" bis hin zu "*good luck*", sind aber nur im »geschwätzigen« Modus (Option -v) zu sehen.

-A Der Additional-Modus besteht aus einer Kombination der Schalter `-O` und `-sV`; eine gute Wahl für alle Anwender, die mehr Wert auf umfangreiche Informationen denn auf eine rasche Analyse legen.

-v Der *verbose*-Schalter erhöht die Geschwätzigkeit von `nmap` und ergänzt die Ausgabe um einige Detailinformationen.

-p Sollen nur bestimmte Ports analysiert werden, kann dies mir der Option `-p` festgelegt werden. Eine Angabe wie »`-p 20-30,6000`« würde alle Ports von 20 bis 30 sowie Port 6000 testen.

6.2.3 Beispiele

Zur Veranschaulichung unterschiedlicher Verfahren und Informationsausgaben zeigen wir Ihnen hier einige beispielhafte `nmap`-Durchläufe.

Zunächst eine Betriebssystemanalyse einer Windows-XP-Workstation:

```
$ sudo nmap -A 192.168.178.210

Starting Nmap 5.21 ( http://nmap.org ) at 2011-10-03 18:28 CEST
Nmap scan report for 192.168.178.210
Host is up (0.00033s latency).
All 1000 scanned ports on 192.168.178.210 are filtered
MAC Address: 08:00:27:6C:E6:50 (Cadmus Computer Systems)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

HOP RTT     ADDRESS
1   0.33 ms 192.168.178.210

OS and Service detection performed. Please report any incorrect▷
 ◁ results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.43 seconds
```

Dieser Rechner ist nicht in eine Domäne integriert und bietet keine Serverdienste an. Entsprechend findet `nmap` auch nichts Interessantes.

Als nächstes eine Betriebssystemanalyse einer FRITZ!Box Fon WLAN 7390, so wie sie aus dem Internet zu sehen ist. Der offene HTTPS-Port wird für die Fernwartung gebraucht:

```
$ sudo nmap -O -v 217.226.237.239

Starting Nmap 5.21 ( http://nmap.org ) at 2011-10-03 18:11 CEST
Initiating Ping Scan at 18:11
Scanning 217.226.237.239 [4 ports]
Completed Ping Scan at 18:11, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 18:11
Completed Parallel DNS resolution of 1 host. at 18:11, 0.00s elapsed
Initiating SYN Stealth Scan at 18:11
Scanning pD9E2EDEF.dip.t-dialin.net (217.226.237.239) [1000 ports]
Discovered open port 443/tcp on 217.226.237.239
Completed SYN Stealth Scan at 18:11, 4.75s elapsed (1000 total ports)
Initiating OS detection (try #1) against pD9E2EDEF.dip.t-dialin.net▷
 ◁ (217.226.237.239)
Retrying OS detection (try #2) against pD9E2EDEF.dip.t-dialin.net▷
 ◁ (217.226.237.239)
Nmap scan report for pD9E2EDEF.dip.t-dialin.net (217.226.237.239)
Host is up (0.0065s latency).
Not shown: 998 filtered ports
```

```
PORT     STATE  SERVICE
443/tcp  open   https
8089/tcp closed unknown
Device type: general purpose|WAP|broadband router|proxy server|▷
 ◁media device|router
Running (JUST GUESSING) : Linux 2.6.X|2.4.X (94%),▷
 ◁ Gemtek embedded (90%), Siemens embedded (90%), Aastra embedded (88%),▷
 ◁ SonicWALL embedded (88%), Chumby embedded (87%),▷
 ◁ MikroTik RouterOS 2.X (87%), Belkin Linux 2.4.X (86%)
Aggressive OS guesses: Linux 2.6.23 (Gentoo) (94%), Linux 2.6.18 (91%),▷
 ◁ Linux 2.6.18 (Debian, x86) (91%), Gemtek P360 WAP or Siemens▷
 ◁ Gigaset SE515dsl wireless broadband router (90%),▷
 ◁ Linux 2.6.15 - 2.6.27 (90%),▷◁ Linux 2.6.13 - 2.6.19 (89%),▷
 ◁ Linux 2.6.13 - 2.6.28 (89%), Linux 2.6.15 - 2.6.28 (89%),▷
 ◁ Linux 2.6.18 - 2.6.24 (89%), Linux 2.6.18 - 2.6.26 (89%)
No exact OS matches for host (test conditions non-ideal).
Uptime guess: 5.064 days (since Wed Sep 28 16:40:04 2011)
TCP Sequence Prediction: Difficulty=193 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/share/nmap
OS detection performed. Please report any incorrect results▷
 ◁ at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.48 seconds
           Raw packets sent: 2039 (92.920KB) | Rcvd: 44 (2732B)
```

Zum Schluss eine Versionsanalyse eines Debian-Systems:

```
$ sudo nmap -A 192.168.178.130

Starting Nmap 5.21 ( http://nmap.org ) at 2011-10-03 18:04 CEST
Nmap scan report for ceol-eth.fritz.box (192.168.178.130)
Host is up (0.000047s latency).
Not shown: 993 closed ports
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 5.9p1 Debian 1 (protocol 2.0)
| ssh-hostkey: 1024 4d:7b:be:ae:18:5a:19:b2:60:05:3a:c4:43:53:d5:e5 ▷
 ◁ (DSA)
|_2048 8c:ca:88:e8:91:c1:66:ec:4f:5d:8b:a5:0b:a8:ba:14 (RSA)
53/tcp   open  domain   dnsmasq 2.58
80/tcp   open  http     Apache httpd 2.2.20 ((Debian))
|_html-title: Site doesn't have a title (text/html).
143/tcp  open  imap     Dovecot imapd
|_imap-capabilities: LOGIN-REFERRALS STARTTLS IMAP4rev1 ENABLE▷
 ◁ AUTH=PLAIN LITERAL+ IDLE SASL-IR ID
443/tcp  open  ssl/http Apache httpd 2.2.20 ((Debian))
|_html-title: Requested resource was http://ceol-eth.fritz.box:443/▷
 ◁ and no page was returned.
993/tcp  open  ssl/imap Dovecot imapd
|_imap-capabilities: IMAP4rev1 AUTH=PLAIN ENABLE ID LITERAL+ IDLE▷
 ◁ SASL-IR LOGIN-REFERRALS
No exact OS matches for host (If you know what OS is running on it,▷
 ◁ see http://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=5.21%D=10/3%OT=22%CT=1%CU=31568%PV=Y%DS=0%DC=L%G=Y▷
 ◁%TM=4E89DD72%P=
◁◁◁◁◁
Network Distance: 0 hops
Service Info: OS: Linux
```

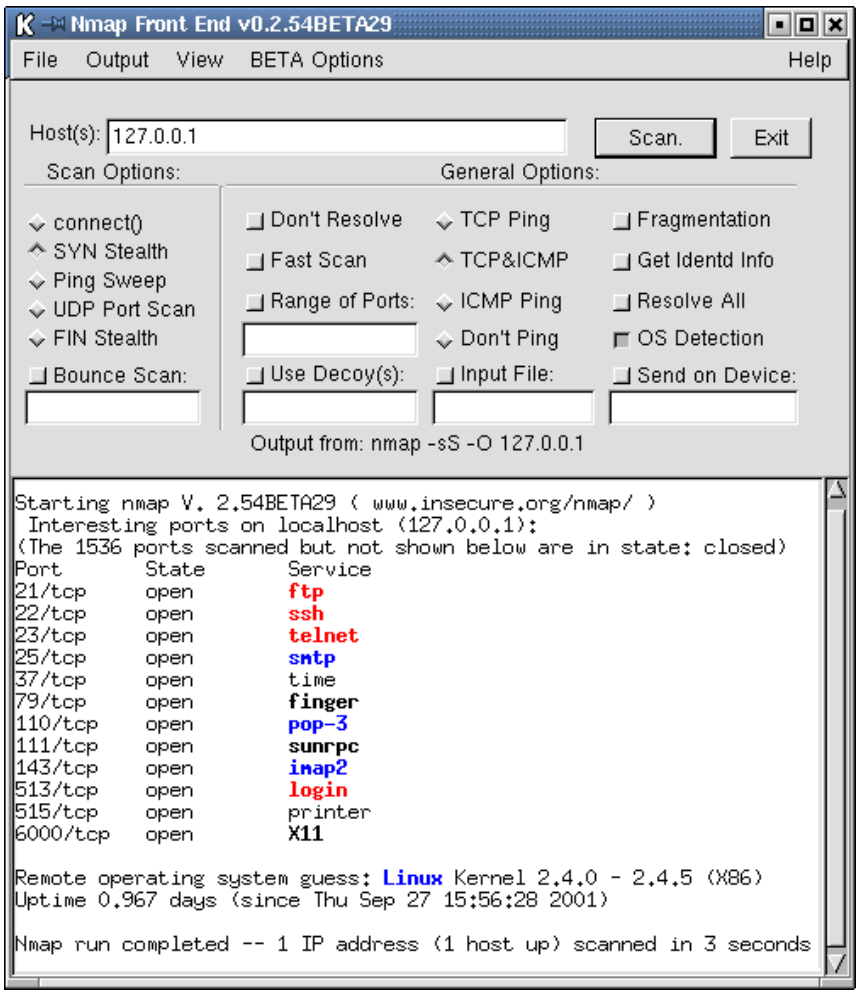


Bild 6.1: `xnmap`, ein grafisches Frontend für `nmap`

```
OS and Service detection performed. Please report any incorrect▷
 ◁ results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 100.56 seconds
```

Wie in Bild 6.1 zu sehen ist, besitzt `nmap` auch ein Frontend für die grafische Oberfläche. Dieses wird mit dem Kommando `xnmap` aufgerufen und erlaubt die Auswahl verschiedener Aktivitäten, ohne die genauen Kommandozeilenparameter kennen zu müssen.

Übungen

6.2 [!3] Analysieren Sie Ihren Rechner mit `nmap` einmal auf der Adresse `127.0.0.1`, dann auf der »richtigen« IP-Adresse. Sehen Sie einen Unterschied und wenn ja, warum?

6.3 [2] Stellen Sie fest, welches Betriebssystem auf dem Standardgateway Ihres lokalen Netzes (falls vorhanden) installiert ist.

6.4 [3] Auch FTP-Server lassen sich ggf. als Vehikel für Scans missbrauchen. Suchen Sie die zugehörige Information aus der Handbuchseite von `nmap` heraus und diskutieren Sie dieses Verfahren.

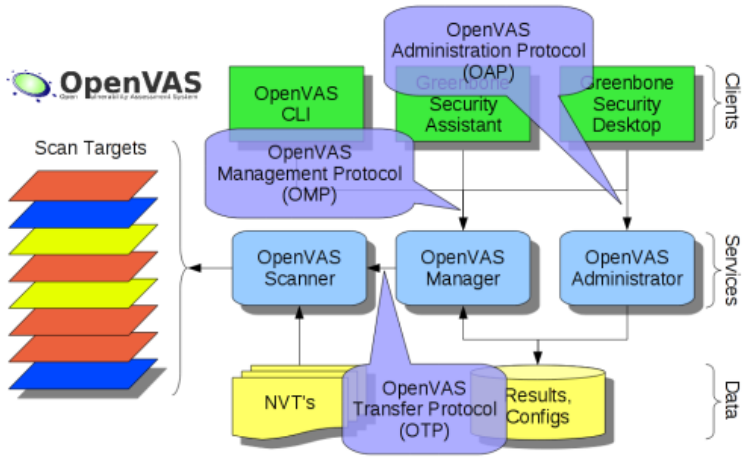


Bild 6.2: Struktur von OpenVAS (Quelle: `http://www.openvas.org/`)

6.3 Der Sicherheitsscanner OpenVAS

6.3.1 Einleitung

Neben reinen Portscannern wie etwa `nmap` stehen Ihnen auch mächtigere Werkzeuge zur Sicherheitsanalyse bereit. Dazu gehören zum Beispiel Programme, die potentielle Schwachstellen eines Systems aufzudecken versuchen. Diese können dann hoffentlich rechtzeitig behoben werden, bevor jemand sie für einen Angriff ausnutzt. Ein prominenter frei verfügbarer Vertreter dieser Programmgattung ist OpenVAS, selbst eine weiterentwickelte Version des Programms Nessus, das 2005 von einer freien auf eine proprietäre Lizenz umgestellt wurde. OpenVAS basiert auf der letzten frei verfügbaren Version von Nessus, hat in der Zwischenzeit aber diverse Veränderungen und Erweiterungen erfahren. Kommerzielle Unterstützung für OpenVAS ist ebenfalls verfügbar.

OpenVAS findet sich im Netz auf `http://www.openvas.org/`. Es ist in den wichtigen Linux-Distributionen enthalten; wie man an aktuelle Versionen für die meisten Distributionen kommt, verrät die OpenVAS-Webseite. Dort gibt es OpenVAS auch als vorinstallierte »virtuelle Maschine« für VirtualBox bzw. VMware, mit dem Sie OpenVAS ausprobieren können, ohne es selbst installieren zu müssen.

Systeme wie OpenVAS müssen für maximale Effektivität – ähnlich wie Virenscanner für Windows – ständig mit aktuellen Informationen über neu entdeckte Sicherheitslücken »gefüttert« werden. OpenVAS kann automatisch einen täglich aktualisierten »Feed« nutzen, der den derzeit über 20.000 Tests für Sicherheitsprobleme bei Bedarf neue hinzufügt.

6.3.2 Struktur

Das OpenVAS-System besteht aus mehreren interagierenden Komponenten:

- Der **OpenVAS-Scanner** kümmert sich um die tatsächliche Ausführung von Sicherheitstests, um das oder die Zielsystem(e) zu untersuchen. OpenVAS-Scanner
- Der Scanner wird vom **OpenVAS-Manager** gesteuert, der sozusagen die »Intelligenz« liefert. Er führt auch eine Datenbank der gefundenen Probleme und bereitet die Resultate auf. OpenVAS-Manager
- Verschiedene Benutzungsoberflächen geben Ihnen Zugriff auf das OpenVAS-System:

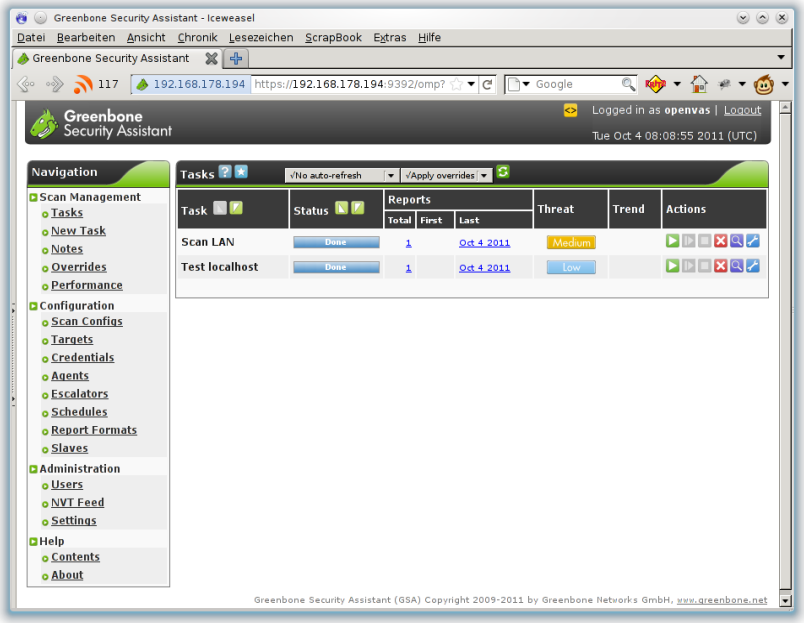


Bild 6.3: Der »Greenbone Security Assistant«

OpenVAS CLI

- **OpenVAS CLI** stellt ein Programm namens `omp` zur Verfügung, mit dem Sie wie in der Linux-Shell textorientierte Kommandos verwenden können.
- Der »Greenbone Security Assistant« ist eine Oberfläche für OpenVAS, die in einem Web-Browser läuft. Sie besteht im Wesentlichen aus einem kleinen Web-Server, der – wie auch das OpenVAS CLI – mit dem OpenVAS-Manager über ein dediziertes Protokoll, das »OpenVAS Management Protocol« (OMP) kommunizieren kann.
- Der »Greenbone Security Desktop« ist ein Qt-basiertes Client-Programm. Es verwendet ebenfalls OMP und steht für Linux, Windows und andere Betriebssysteme zur Verfügung.

OpenVAS Administrator

- Verwaltungsaufgaben übernimmt der **OpenVAS Administrator**, der sich vor allem um Benutzerrechte und die Administration des Feeds kümmert. Die Oberflächen gestatten privilegierten Benutzern den Zugriff auf den OpenVAS Administrator über das »OpenVAS Administration Protocol« (OAP).

Nur falls Sie sich fragen: Greenbone (`http://www.greenbone.net/`) ist ein in Osnabrück ansässiges Unternehmen, das Produkte und Dienstleistungen rund um OpenVAS anbietet.

Der OpenVAS Manager und der OpenVAS Scanner kommunizieren untereinander über das »OpenVAS Transfer Protocol« (OTP). Aus der Nessus-Ära übriggeblieben ist ein »alter« Client, der ebenfalls OTP verwendet. Allerdings wird dessen Benutzung ebenso wie die von OTP direkt nicht mehr empfohlen, da das Projekt darüber nachdenkt, das Protokoll durch etwas Anderes zu ersetzen.

Diese flexible Struktur erlaubt es, mehrere Zielsysteme gleichzeitig zu analysieren und die Arbeit dafür über verschiedene Rechner zu verteilen.

6.3.3 OpenVAS benutzen

Die bequemste Möglichkeit, OpenVAS zu benutzen, geht über den »Greenbone Security Assistant«, also die Web-Oberfläche (Bild 6.3). Wenn Sie auf demselben

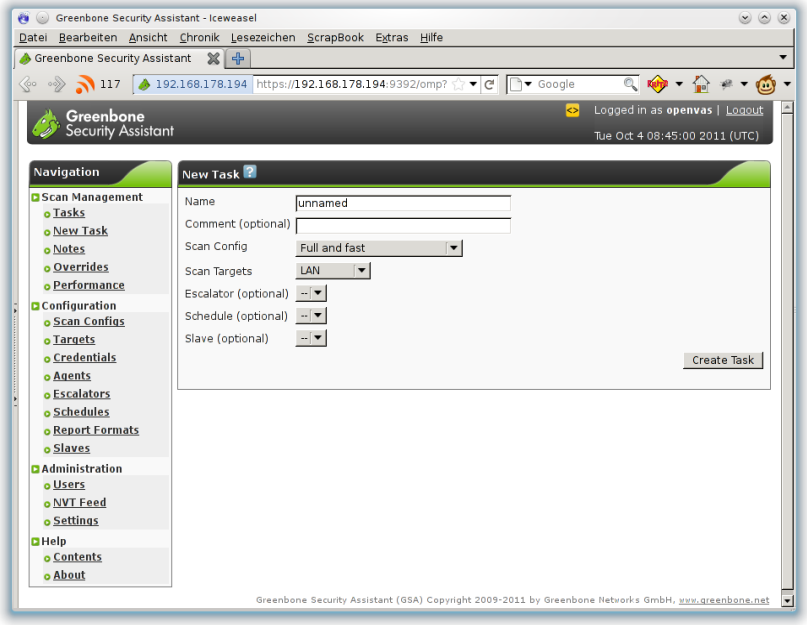


Bild 6.4: Neue OpenVAS-Task anlegen

Rechner angemeldet sind, auf dem das Programm läuft, erreichen Sie es über `https://localhost:9392/` (beachten Sie das »`https`«).

Wenn Sie zum Testen die virtuelle Maschine von `www.openvas.org` verwenden, dann bekommen Sie den URL für den Zugang nach deren Start angezeigt.

Je nachdem, welchen Browser Sie verwenden, könnte es sein, dass Sie eine Fehlermeldung bekommen, weil die Verbindung kein von einer anerkannten Zertifizierungsstelle signiertes Zertifikat verwendet. Tun Sie, was Sie tun müssen, um das Zertifikat trotzdem zu akzeptieren.

Als erstes müssen Sie einen Benutzernamen und ein Kennwort angeben. Sie sollten entweder bei der OpenVAS-Installation darum gebeten worden sein, Benutzerdaten einzugeben – in diesem Fall benutzen Sie die, die Sie dort eingegeben haben –, oder (bei der virtuellen Maschine) Sie verwenden die dort nach dem Start angezeigten Benutzerdaten. Nach dem Anmelden sollte ein Bildschirm ähnlich dem in Bild 6.3 erscheinen.

Analysieren

Das Analysieren eines oder mehrerer Zielsysteme veranlassen Sie in OpenVAS über eine *task* (Aufgabe). Wählen Sie hierzu aus der Menüspalte am linken Fensterrand den Eintrag *New Task*. Danach erscheint die in Bild 6.4 gezeigte Maske. Hier können Sie der Task einen Namen geben sowie einen Kommentar hinzufügen, auswählen, welche Prüfungen durchgeführt werden sollen (*Scan Config*) und welche Rechner einbezogen werden sollen (*Scan Targets*).

New Task

Die übrigen drei Menüs sind eher für fortgeschrittene Anwender: *Escalator* erlaubt die Definition von automatischen Reaktionen auf Analyseergebnisse – zum Beispiel könnte OpenVAS Ihnen automatisch eine Mail schicken, wenn auf einem Rechner wichtige Sicherheitsprobleme gefunden wurden. *Schedule* gestattet es, die Analyse periodisch zu wiederholen. *Slave* schließlich erlaubt es, einen OpenVAS-Scanner auf einem anderen Rechner als dem lokalen die eigentliche Analyse durchführen zu lassen. Die Einträge *Escalators*, *Schedules* und *Slaves* in der Menüspalte am linken Fensterrand dienen dazu, Einträge für diese Menüs festzulegen.

Im Menü *Scan Config* gibt es die folgenden Vorgaben:

Scan Config

Full and fast führt alle Prüfungen durch, die die Stabilität eines zu analysierenden Rechners nicht gefährden (manche Tests könnten Dienste auf einem Rechner oder gar den Rechner selbst unter bestimmten Umständen abstürzen lassen). Dabei greift OpenVAS, wo möglich, auf abgespeicherte Ergebnisse früherer Analyseläufe zurück, um Zeit zu sparen.

Full and fast ultimate entspricht der vorigen Option, bis darauf, dass auch die Tests mit Absturzgefahr vorgenommen werden.

Full and very deep entspricht der ersten Option, bis darauf, dass OpenVAS sich nicht auf frühere Ergebnisse verläßt.

Full and very deep ultimate ist eine Kombination der beiden vorstehenden Optionen.

Sie können über den Eintrag *Scan Configs* in der Menüspalte Ihre eigenen Vorstellungen realisieren. Dazu müssen Sie im oberen Teil der Maske einen Namen und ggf. einen Kommentar eingeben und *Create Scan Config* auswählen. Als Basis für Ihre eigene Konfiguration können Sie entweder auf eine leere Konfiguration oder auf *Full and fast* zurückgreifen.

Im Prinzip können Sie jeden der über 20.000 Tests einzeln ein- und ausschalten. Das »Schraubenschlüssel«-Icon in der Zeile einer Konfiguration unter *Scan Configs* (im unteren Teil der Maske) gibt Ihnen zunächst Zugriff auf die »Familien« von Tests und dann innerhalb der Familien auf die einzelnen Tests, mit umfangreichen Erklärungen.

Sie können bei der Analyse Zeit sparen, indem Sie für die einzelnen Zielrechner nur diejenigen Tests ausführen, die auch Sinn ergeben. Zum Beispiel bringt es nichts, einen Linux-Rechner nach Windows-Sicherheitslücken abzuklopfen oder einen Router nach Problemen mit der Oracle-Datenbank.

NASL

Die Tests selbst werden in einer speziellen Programmiersprache namens NASL (»Nessus Attack Scripting Language«) geschrieben und stehen (typischerweise) unter `/var/lib/openvas` zur Verfügung. Sie können sich dort umschauen und bei Bedarf auch Ihre eigenen Skripte erstellen.

Targets Im Menü *Targets* können Sie bestimmen, welche Rechner oder Netze analysiert werden können. Was Sie hier festlegen, taucht später im *Scan Targets*-Menü von *New Task* auf:

Name und *Comment* sprechen für sich selbst.

Hosts gibt an, welche Rechner oder Netze enthalten sein sollen. Hier können Sie eine durch Komma getrennte Folge von Einträgen angeben, wobei jeder Eintrag entweder ein Rechnername, die IP-Adresse eines Rechners, die IP-Adresse eines Netzes (mit Netzmaske) oder ein IP-Adressenbereich sein kann – die genaue Syntax können Sie erfahren, indem Sie auf das Fragezeichen-Icon hinter dem Titel *New Target* klicken.

Port Range gibt die Ports an, die bei der Analyse angeschaut werden. Außer *default* kann dort eine durch Komma getrennte Folge von Portnummern oder Portnummer-Bereichen (mit Minuszeichen getrennt) stehen.

Mit *SSH Credential* und *SMB Credential* können Sie Zugangsdaten für die Secure Shell und SMB (vulgo Samba) angeben, die OpenVAS ausnutzt, um noch ausführlichere Prüfungen vorzunehmen. Zum Beispiel kann es über SSH die Paketliste eines Linux-Rechners abrufen und nach als unsicher bekannten Programmversionen schauen. Die entsprechenden Menüs beschicken Sie über die *Credentials*-Seite.

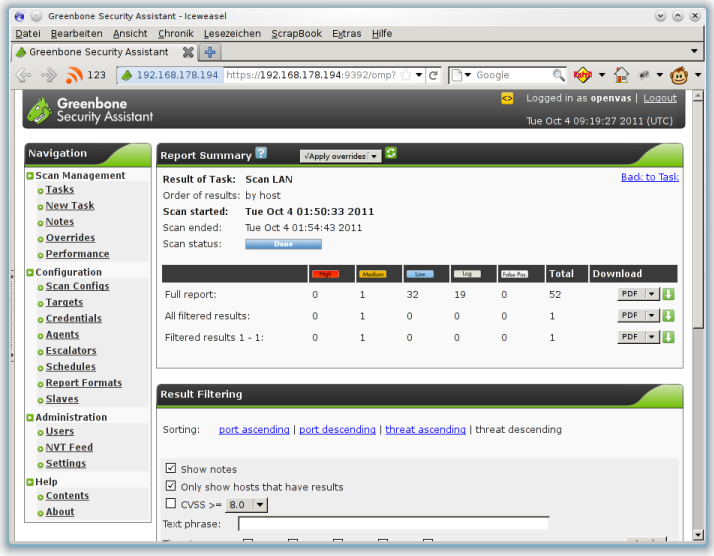


Bild 6.5: OpenVAS-Analyse-Ergebnis

Wenn Sie die gewünschte Analyse konfiguriert haben (mit *New Task* unter Zuhilfenahme von *Targets* und gegebenenfalls *Scan Configs*), können Sie auf der *Tasks*-Seite (Bild 6.3) den Analysevorgang starten. Lokalisieren Sie dazu die Zeile in der Task-Liste, die Ihrer Analyse entspricht, und klicken Sie auf das Pfeil-Icon in der rechten Spalte, *Actions* (denken Sie an Ihren CD-Player oder Videorecorder). Analysevorgang starten

Als nächstes erscheint unter *Status* zunächst die Meldung *Requested*. Danach passiert anscheinend gar nichts, aber das täuscht – klicken Sie auf das grüne Icon mit den beiden gekrümmten Pfeilen rechts neben den Menüs *No auto-refresh* und *Apply overrides*, um die Statusangabe zu aktualisieren.

Wenn Sie im Menü *No auto-refresh* erst zum Beispiel *Refresh every 10 Sec.* auswählen und dann auf das grüne Icon rechts davon klicken, wird der Status automatisch alle zehn Sekunden aktualisiert.

Wenn die Analyse abgeschlossen ist, zeigt die Statusspalte *Done*, und unter *Reports* erscheint ein Verweis auf den neuesten Report (bei *Last*). Wenn Sie diesen Verweis anklicken, bekommen Sie eine Seite ähnlich der in Bild 6.5.

Die Ergebnisseite gibt Ihnen zunächst einen Überblick über die gefundenen Ergebnisse. OpenVAS unterscheidet dabei zwischen Bedrohungen hoher, mittlerer und niedriger Stufe und Protokolleinträgen, die über den Ablauf der Analyse berichten. Sie haben Zugriff auf den kompletten Bericht (*Full report*) oder gefilterte Versionen, wobei Sie die Filterung im unteren Teil der Maske konfigurieren können. Sie können sich zum Beispiel bei der Analyse eines Netzes im Ergebnis auf diejenigen Stationen beschränken, bei denen tatsächlich etwas Bemerkenswertes gefunden wurde. Ergebnisseite

Das Ergebnis können Sie in verschiedenen Formaten abrufen, etwa als PDF, XML, HTML, einfachem Text oder in diversen kommaseparierten Formaten, die Sie zum Beispiel in einer Tabellenkalkulation importieren könnten.

Um die PDF-Ausgabe erhalten zu können, müssen Sie LaTeX installiert haben.

Bild 6.6 zeigt den Anfang eines HTML-Berichts über einen Rechner.

Zur skriptgesteuerten Weiterverarbeitung von OpenVAS-Ergebnissen eignet sich zum Beispiel das »Nessus-Backend-Format« NBE. Für jede entdeckte Problemstelle wird eine eigene Textzeile erzeugt und die Informationen dort in der allgemeinen Form

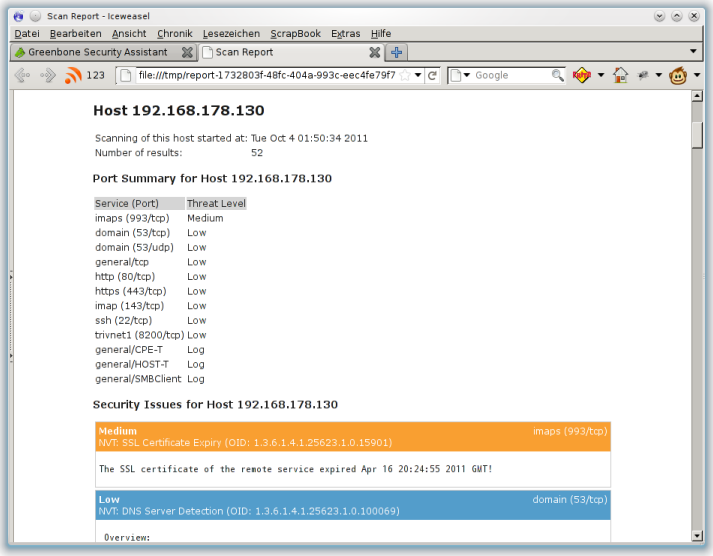


Bild 6.6: OpenVAS-Ergebnisbericht

```
timestamps|||scan_start|Tue Oct  4 01:50:33 2011|
results|192.168.178.130|192.168.178.130|imaps (993/tcp)▷
 ◁|1.3.6.1.4.1.25623.1.0.15901|Security Warning|The SSL certificate▷
 ◁ of the remote service expired Apr 16 20:24:55 2011 GMT!\n
results|192.168.178.130|192.168.178.130|domain (53/tcp)▷
 ◁|1.3.6.1.4.1.25623.1.0.100069|Security Note|\n\n Overview:▷
 ◁\n  A DNS Server is running at this Host.\n ▷
 ◁ A Name Server translates domain names into IP addresses. This▷
 ◁ makes it\n  possible for a user to access a website▷
 ◁ by typing in the domain name instead of\n  the website's▷
 ◁ actual IP address.\n\n Risk factor : None\n
◁◁◁◁◁
```

Bild 6.7: Auszug aus einem Nessus-Bericht im NBE-Format

`results|`⟨*Scanner*⟩`|`⟨*Rechner*⟩`|`⟨*Port*⟩`|`⟨*ID*⟩`|`⟨*Typ*⟩`|`⟨*Bericht*⟩

abgelegt. Dabei stellt die ⟨*ID*⟩ die Nummer des verwendeten Plugins dar, während der ⟨*Typ*⟩ anzeigt, um welche Art von Feststellung (Problem, Warnung oder Hinweis) es sich handelt. ⟨*Scanner*⟩ ist der Rechner, auf dem der Scanner lief, und ⟨*Rechner*⟩ und ⟨*Port*⟩ beschreiben die Station, wo das Problem bemerkt wurde. Ein Beispiel sehen Sie in Bild 6.7.

Übungen

6.5 [3] Installieren und konfigurieren Sie OpenVAS. Lesen Sie gegebenenfalls die Dokumentation, die Ihre Distribution mitbringt. (Wenn OpenVAS nicht in Ihrer Distribution enthalten ist, können Sie auch die virtuelle Maschine verwenden.)

6.6 [!3] Verwenden Sie OpenVAS, um Ihren eigenen oder den Rechner eines anderen Schulungsteilnehmers zu untersuchen. Finden Sie gravierende Sicherheitslöcher?

Kommandos in diesem Kapitel

`nmap`	Netzwerk-Portscanner, analysiert offene Ports auf Rechnern	`nmap`(1)	90
`omp`	Textorientierte Oberfläche für OpenVAS	`omp`(8)	97
`xnmap`	Grafisches Frontend für `nmap`	`xnmap`(1)	96

Zusammenfassung

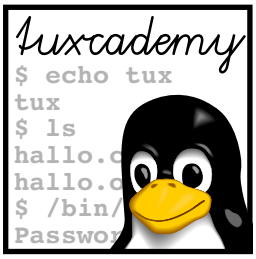
- Sicherheitsscanner gibt es als freie und als proprietäre Programme. Die Existenz freier Sicherheitsscanner ist nicht unumstritten.
- `nmap` ist ein Portscanner, der von außen prüft, welche Dienste ein Rechner anbietet. Er kann auch versuchen, das Betriebssystem zu erkennen sowie die Versionen verschiedener Serverprogramme zu bestimmen.
- OpenVAS ist ein umfangreicher Client-Server-orientierter Sicherheitsscanner. Es enthält eine Datenbank mit zahlreichen Programmen, die die Existenz von Sicherheitslücken nachzuweisen versuchen. OpenVAS liefert umfangreiche Informationen über gefundene Sicherheitslücken und darüber, wie schwerwiegend diese sind.

Literaturverzeichnis

Fyo98 Fyodor. »Remote OS detection via TCP/IP Stack Fingerprinting«, Oktober 1998. `http://insecure.org/nmap/nmap-fingerprinting-article.txt`

Fyo02 Fyodor. »Idle Scanning and Related IPID Games«, September 2002. `http://www.insecure.org/nmap/idlescan.html`

RFC0793 Information Sciences Institute. »Transmission Control Protocol«, September 1981. `http://www.ietf.org/rfc/rfc0793.txt`



7

Rechnerbasierte Angriffserkennung

Inhalt

Lernziele

- Techniken zur rechnerbasierten Angriffserkennung kennen
- Die Programme Tripwire und AIDE konfigurieren und einsetzen können

Vorkenntnisse

- Kenntnisse der Administration von Linux-Systemen

7.1 Einleitung

Trotz aller Abwehrmaßnahmen passiert es manchmal doch: Die Cracker dringen in einen Ihrer Rechner ein und bringen ihn in ihre Gewalt. Meistens äußert sich das heutzutage nicht mehr darin, dass sie mit einem hämischen Kichern die Festplatte löschen. Statt dessen wird viel öfter Software installiert, die den Rechner zum Teil eines »Botnetzes« machen soll, damit er sich für seine neuen Oberherrscher nützlich macht – etwa durch Spamversand oder das Angreifen anderer Rechner. (Natürlich kriegen *Sie* dafür dann den Ärger.)

Die Frage ist also: Wie können Sie als Systemadministrator solche Machenschaften erkennen und die Eindringlinge vertreiben? Ersteres ist die Domäne
rechnerbasierte IDS **rechnerbasierter Angriffserkennungssysteme** (engl. *host-based intrusion detection systems* oder *IDS*). Das Problem dabei ist natürlich, dass Sie versuchen müssen, herauszufinden, was auf dem Rechner Sache ist, aber Sie dem Rechner eigentlich nicht mehr vertrauen können – die Cracker versuchen natürlich, ihre Spuren zu
Rootkit verwischen. Gängige »Rootkits«, also Softwarepakete, die Eindringlinge nutzen, um sich einen dauerhaften Zugang zu Ihrem Rechner zu verschaffen, können zum Beispiel dafür sorgen, dass ihre Prozesse und Dateien in der Ausgabe von `ps` und `ls` nicht mehr auftauchen, so dass sie um so schwieriger zu finden sind.

Auch wenn betroffene Endbenutzer das in der Regel ungern hören: Ist ein Rechner erst einmal kompromittiert, so führt an einer kompletten Neuinstallation normalerweise kein Weg vorbei[1]. Damit ist der Hauptzweck der Rechnerbasierten Angriffserkennung nicht so sehr das Wiederherstellen des Systems, sondern vielmehr überhaupt das Erkennen eines erfolgten Angriffs.

Prinzipiell gibt es zwei Möglichkeiten, eine Veränderung am System nachträglich festzustellen: entweder Sie wissen, wie ein sauberes System aussieht, oder Sie wissen, wie ein verändertes System aussieht. Im ersten Fall benötigen Sie eine
Datenbank möglichst vollständige Bestandsaufnahme eines sauberen Systems, eine Datenbank des Systemzustands. Da sich Systeme im Verlauf ihrer Lebenszeit verändern, ist es für eine erfolgreiche Angriffserkennung notwendig, diese Datenbank aktuell zu halten, was sehr aufwendig sein kann.

Im zweiten Fall brauchen Sie eine möglichst vollständige Bestandsaufnahme aller Schändlichkeiten, die jemand mit Ihrem System anstellen kann. Da nicht schon marginale Variationen den Detektor aushebeln sollen, wird hier vor allem
Signaturen mit Signaturen gearbeitet. Prinzipbedingt ist die zweite Variante nicht so zuverlässig: weder kann man heute schon wissen, was morgen für eine neue Lücke entdeckt wird[2], noch lässt sich immer verhindern, das es Unschuldige trifft: ein eigentlich sauberes Programm, das zufällig auf die Signatur eines Schädlings passt.

Die von ihrer Funktion her sehr ähnlichen Programme Tripwire und AIDE versuchen, einen Angriff dadurch zu erkennen, dass sie Veränderungen am Dateisystem registrieren. Da beide dafür den Ist-Zustand gegen einen in einer Datenbank hinterlegten Soll-Zustand abgleichen, kommen sie immer erst *nach* einem Angriff zum Tragen; sie sind also nicht proaktiv.

Zu Vertretern der zweiten, signaturbasierten Vorgehensweise gehören Rootkit-Detektoren und Virenscanner.

Übungen

7.1 [2] Warum sollte ».« niemals im Suchpfad von `root` liegen?

7.2 [2] Warum reicht es nach einem möglichen Angriff nicht aus, nur nach *veränderten* Dateien und Programmen Ausschau zu halten?

7.3 [3] Welche Programme wird ein Eindringling möglicherweise verändern, um seine Anwesenheit zu verschleiern?

[1] Selbst wenn man davon ausginge, die zahlreich feilgebotenen Werkzeuge könnten wirklich den Eindringling und alle seine Hintertüren restlos entfernen, so bleibt der Rechner doch ein »wirtschaftlicher Totalschaden«: das wirklich restlose Entfernen aller Hinterlassenschaften ist aufwendiger als eine Neuinstallation und das Zurückspielen der Daten aus einer Sicherheitskopie.

[2] Das ist der Grund, warum Viren-Scanner nicht komplett vor Viren-Befall schützen.

7.2 Tripwire

7.2.1 Aufbau

Tripwire (engl. *Stolperdraht*) ist ein Datei-Integritätstester, der versucht Manipulation an Dateien aufgrund ihrer Abweichung gegenüber dem Soll-Zustand zu erkennen. Er wurde sowohl kommerziell vermarktet (von Tripwire, Inc.) als auch unter der GPL als freies Projekt weitergeführt (`http://sourceforge.net/projects/tripwire/`).

Das Tripwire-Paket besteht aus dem Programm `tripwire` und mehreren Hilfsprogrammen, von denen inbesondere `twadmin` und `twprint` hervorzuheben sind. Die Programme operieren mit vier Dateien: die eigentliche Konfigurationsdatei, die normalerweise nicht verändert werden muss, da sie nur Pfadangaben u. Ä. enthält; eine Datei, die die Überwachungsrichtlinien enthält; die Referenz-Datenbank (Soll-Zustand) und eine oder mehrere Berichts-Dateien (aktueller Ist-Zustand). Konfigurationsdatei und Richtliniendatei müssen für den Betrieb mit Tripwire in ein Binärformat umgewandelt werden; diese Dateien und die Referenz-Datenbank werden zudem signiert. Mehr über die einzelnen Bestandteile von Tripwire verrät `twintro`(8).

7.2.2 Vorbereitende Arbeiten

Bevor Sie Tripwire benutzen können, müssen Sie zwei Schlüssel erstellen: den globalen Schlüssel (*site key*) und den lokalen Schlüssel (*local key*). Tripwire benutzt den globalen Schlüssel zum Signieren der Dateien, die für alle Rechner gleich sind, d. h. die Konfigurations- und Richtliniendatei. Mit dem lokalen Schlüssel wird die Datenbank signiert. Für die Schlüsselgenerierung ist `twadmin` zuständig:

globaler Schlüssel
lokaler Schlüssel

```
# cd /etc/tripwire
# twadmin --generate-keys --site-keyfile site.key
# twadmin --generate-keys --local-keyfile $(hostname)-local.key
```

Dabei erzeugt `twadmin` nicht nur jeweils den Schlüssel, sondern fragt Sie auch nach einem Kennwort, um diesen zu sichern. Sollten Sie mehrere Rechner überwachen wollen, so kopieren Sie `site.key` auf alle diese Rechner und wiederholen dort jeweils den letzten Schritt, um einen lokalen Schlüssel zu erzeugen.

Als nächstes müssen Sie die Konfigurationsdatei erstellen. Diese muss minimal die folgenden Variablen enthalten:

```
# cat twcfg.txt
POLFILE         = /etc/tripwire/tw.pol
DBFILE          = /var/lib/tripwire/$(HOSTNAME).twd
REPORTFILE      = /var/lib/tripwire/report/$(HOSTNAME)-$(DATE).twr
SITEKEYFILE     = /etc/tripwire/site.key
LOCALKEYFILE    = /etc/tripwire/$(HOSTNAME)-local.key
EDITOR          = /bin/vi
```

Dabei müssen die Werte von `SITEKEYFILE` und `LOCALKEYFILE` natürlich zu den Pfaden für Ihren globalen bzw. lokalen Schlüssel passen. Die Werte von `POLFILE`, `DBFILE` und `REPORTFILE` bestimmen die Lage der Richtliniendatei, der Referenz-Datenbank bzw. der Berichte. Die Verwendung der Variable `HOSTNAME` – wie auch die Benutzung des `hostname`-Kommandos oben – setzt natürlich eine korrekt konfigurierte (lokale) Namensauflösung voraus. Die Syntax und Bedeutung weiterer Variablen erklärt die Handbuchseite `twconfig`(4), über die verschiedenen Dateien erfahren Sie mehr in `twfiles`(5).

Die eigentlichen Überwachungsrichtlinien werden in der Datei `twpol.txt` festgehalten. Wir beschränken uns hier auf ein simples Beispiel; im Detail gehen wir auf die Richtlinien in Abschnitt 7.2.4 ein.

```
# cat twpol.txt
/usr          ->          $(ReadOnly);
```

(Beachten Sie das Semikolon am Ende.) Diese Richtlinie besagt, dass das Verzeichnis `/usr` und alle darin enthaltenen Dateien, Verzeichnisse, Unterverzeichnisse usw. überwacht werden sollen, wobei davon ausgegangen wird, dass sich die Dateien und Verzeichnisse nicht ändern.

Die Klartext-Dateien werden schließlich durch

```
# twadmin --create-cfgfile --site-keyfile site.key twcfg.txt
# twadmin --create-polfile --site-keyfile site.key twpol.txt
```

in die (signierten) Binärdateien `tw.cfg` bzw. `tw.pol` übersetzt. Kopieren Sie diese ggf. auf weitere zu überwachende Rechner.


Mit ein wenig Aufwand können Sie Konfigurations- und Richtliniendatei so gestalten, dass sie nicht vom jeweiligen Rechner abhängen.

7.2.3 Regel-Betrieb

Initialisierung Nachdem die vorbereitenden Arbeiten abgeschlossen sind, können wir nun mit dem Betrieb beginnen. Als erstes muss die Datenbank gemäß den Überwachungsrichtlinien erstellt werden, d. h. der aktuelle Ist-Zustand wird der Soll-Zustand für alle künftigen Überprüfungen: der Referenz-Zustand. Dazu dient das Programm `tripwire`, der Aufruf ist einfach

Referenz-Zustand

```
# tripwire --init
```

Die Datenbank wird an der Stelle abgelegt, die die Variable `DBFILE` in der Konfigurationsdatei bestimmt, und mit dem lokalen Schlüssel signiert. Die Datenbank selbst ist nicht direkt lesbar, Sie können sie aber durch das Programm `twprint` sichtbar machen; nützlich ist hier auch die explizite Angabe einzelner Dateien:

```
# twprint --print-dbfile /etc/passwd
Object name:  /etc/passwd

Property:               Value:
-------------           -----------
Object Type             Regular File
Device Number           777
Inode Number            8900
Mode                    -rw-r--r--
Num Links               1
UID                     root (0)
GID                     root (0)
```


Achten Sie unbedingt darauf, das sich Ihr Rechner bei der Initialisierung in einem *sauberen* Zustand befindet. Am besten richten Sie die Datenbank direkt nach der Installation des Systems ein.

Überprüfung des Systems Die Integrität Ihres Systems können Sie jetzt jederzeit mittels eines Aufrufs von

```
# tripwire --check
```

überprüfen. Dabei schlägt `tripwire` Alarm, wenn Konfigurations- oder Richtliniendatei oder die Datenbank manipuliert wurde und natürlich, wenn eine Datei im

Rahmen der Richtlinien vom Soll-Zustand abweicht. Die Ausgabe von Tripwire informiert Sie in der Standard-Konfiguration jedoch nur, dass eine Datei gegen die Richtlinien verstößt, aber nicht um welchen Verstoß es sich handelt. Bei jeder Überprüfung des Ist-Zustandes wird jedoch ein ausführlicher Bericht erstellt und an der Stelle abgelegt, die die Variable `REPORTFILE` in der Konfigurationsdatei bestimmt. Der Bericht selbst liegt in einem Binärformat vor, Sie können ihn sich jedoch wieder mit `twprint` anschauen. Da in der Standard-Konfiguration der Dateiname des Berichts jedoch einen Zeitstempel enthält, müssen Sie diesen explizit angeben:

```
# twprint --print-report▷
 ◁    --twrfile /var/lib/tripwire/report/⟨Rechner⟩-⟨Zeitstempel⟩.twr
```

Der Aufruf von `tripwire --check` erfordert nicht die Eingabe eines Schlüssels; Sie können Ihn daher problemlos automatisieren, beispielsweise als Cronjob.

Anpassung der Datenbank Während des Betriebs kann es immer vorkommen, dass eigentlich statische Dateien sich ändern, ohne dass ein Angriff stattgefunden hat: beispielsweise nach einem Update von Software oder der Änderung der Konfiguration von Diensten. Natürlich könnten Sie in einem solchen Fall einfach die Datenbank neu initialisieren. Das birgt aber die Gefahr, dass parallel zu den legitimen Änderungen bösartige Veränderungen unentdeckt durchschlüpfen – sind sie erst einmal in der Datenbank eingetragen, können die nicht mehr entdeckt werden.

Es ist deswegen besser, jede Abweichung vom Soll-Zustand einzeln zu begutachten und entweder in die Datenbank zu übernehmen oder genauer zu analysieren. Tripwire bietet Ihnen hierzu eine halbwegs komfortable Unterstützung an. Durch

```
# tripwire --update▷
 ◁    --twrfile /var/lib/tripwire/report/⟨Rechner⟩-⟨Zeitstempel⟩.twr
```

können Sie Einträge aus dem angegebenen Bericht in die Datenbank übernehmen (es wird also immer nur der Ist-Zustand bei der letzten Überprüfung in die Datenbank übernommen, nicht der aktuelle Ist-Zustand des Dateisystems).

Durch den Aufruf des Kommandos wird ein Editor gestartet (welcher das ist, bestimmt die Konfigurations-Variable `EDITOR`); der Dateiinhalt entspricht im Wesentlichen der Ausgabe von `twprint`, allerdings befindet sich am Ende eine Liste der Abweichler mit vorgestellten Kästchen zum Ankreuzen:

```
Remove the "x" from the adjacent box to prevent updating the database
with the new values for this object.

Modified:
[x] "/bin/ls"
[x] "/bin/rm"
```

Wenn Sie nicht wollen, dass der Ist-Zustand in die Datenbank übernommen werden soll, so müssen sie das `x` in dem Kästchen `[x]` vor dem entsprechenden Eintrag entfernen. Alle übrigen Einträge werden nach dem Beenden des Editors und der Eingabe des lokalen Schlüssels in die Datenbank übernommen.

7.2.4 Festlegung der Überwachungsrichtlinien

Struktur der Richtliniendatei Die Richtliniendatei enthält Regeln und Variablen-Zuweisungen (und ggf. noch weitere Einträge, auf die wir hier aber nicht eingehen; Details finden sich in `twpolicy`(4)). Eine Regel hat dabei die Form

```
⟨Dateisystem-Objekt⟩ -> ⟨Tests⟩;
```

also beispielsweise

```
/bin  ->  +ugptsmS;                                  Eigentümer, Rechte und Inhalt
```

die besagt, dass das Verzeichnis `/bin` und alle darin enthaltenen Dateien und Verzeichnisse, Unterverzeichnisse usw. den rechts angegebenen Tests unterzogen werden sollen. (Zu den Tests kommen wir gleich; jeder Test wird durch einen Buchstaben repräsentiert.)

Dabei können Sie für einzelne Dateien oder Verzeichnisse abweichende Regelungen treffen wie in

```
/etc        -> +ugptsmS;                             statische Dateien
/etc/shadow -> +ugpt;                                veränderliche Datei
```

oder durch ein vorangestelltes `!` auch ganz von der Überprüfung ausnehmen:

```
/etc        ->        +ugptsmS;                      Überprüfung
!/etc/opt ;                                          Keine Überprüfung
```

Die rechte Seite einer Regel gibt die durchzuführenden Tests an (jeweils ein Buchstabe); ein `+` bedeutet dabei, das die folgenden Tests durchgeführt werden sollen, ein `-` schaltet alle folgenden Tests ab.

Prinzipiell werden nur Regeln benötigt, aber durch Variablen können Sie die Richtliniendatei übersichtlicher und damit wartbarer gestalten. Eine Variablen-Zuweisung hat die Form

```
⟨Variable⟩ = ⟨Wert⟩;
```

beispielsweise

```
ReadOnly  = +pinugtsdbmCM-rlacSH;
```

Auf diese Variable kann dann über `$(`⟨*Variable*⟩`)` zugegriffen werden:

```
/bin  ->  $(ReadOnly);
```

Tripwire definiert von sich aus einige Variablen, die nicht geändert werden können. Sie enthalten Test-Auswahlen für gängige Fälle: `ReadOnly` (statische Dateien), `Dynamic` (dynamische Dateien), `Growing` (wachsende Dateien, beispielsweise Protokoll-Dateien), `Device` (Gerätedateien), `IgnoreAll` (kein Test; lediglich Abwesenheit oder Existenz wird registriert) und `IgnoreNone` (alles aktiv). Die genauen Einstellungen können Sie der Handbuchseite `twpolicy`(4) entnehmen.

Überwachte Dateieigenschaften Tripwire ist in der Lage, eine ganze Reihe von Tests durchzuführen, deren komplette Aufstellung Sie in Tabelle 7.1 sehen können. Neben den üblichen Dateiattributen wie Eigentümer, Zugriffsrechte, den drei Zeitstempeln oder der Größe kennt Tripwire auch exotischere Daten wie I-Node-Nummer oder Dateisystem-ID (beide Daten zusammen bestimmen eine Datei eindeutig im System unabhängig vom Dateinamen).

Die von Tripwire vordefinierte Variable `ReadOnly` enthält nur die Prüfsummen-Tests `C` und `M`, also CRC-32 und MD5. Ersterer ist aber nicht für die Erkennung bewusster Manipulationen, sondern nur für zufällige Verfälschungen gedacht, etwa nach einem Dateisystem-Crash. Für MD5 hingegen ist es inzwischen in Spezialfällen gelungen, verschiedene Dokumente mit gleicher Prüfsumme zu erzeugen. Leider steht auch SHA nicht mehr so strahlend dar wie noch ehedem. Auch wenn es Rechenzeit kostet sollten Sie deswegen auf Nummer Sicher gehen und alle Prüfsummen aktivieren:

Tabelle 7.1: Tripwire: mögliche Tests von Dateieigenschaften

Test	Bedeutung
`u`	Besitzer (*user*)
`g`	Gruppe (*group*)
`p`	Zugriffsrechte (*permissions*)
`b`	belegte Blöcke
`s`	Dateigröße (*size*)
`l`	wachsende Datei (*log file*)
`a`	Zugriffszeit (*access time*)
`m`	Änderungszeit (*modification time*)
`c`	Metadaten-Änderungszeit (*inode change time*)
`t`	Dateityp (*type*)
`i`	I-Node-Nummer
`n`	Anzahl der Hardlinks
`d`	Dateisystem-ID
`r`	Haupt- und Neben-Gerätenummer
`C`	CRC-32 Prüfsumme
`H`	Haval Prüfsumme
`M`	MD5 Prüfsumme
`S`	SHA Prüfsumme

```
ro = $(ReadOnly) +SH;
/bin -> $(ro);
```


Berechnen Sie niemals Prüfsummen (Tests `C`, `H`, `M` und `S`) von Gerätedateien. Sonst enden Sie damit, Prüfsummen für ganz `/dev/hda` zu berechnen.

Nachträgliche Änderung der Überwachungsrichtlinien Wie auch bei der Datenbank sollten Sie bei einer Änderung der Überwachungsrichtlinien (`twpol.txt`) nicht einfach die zugehörige Binärdatei (`tw.pol`) durch `twadmin` neu anlegen. Vielmehr emphielt es sich, sie durch

```
# tripwire --update-policy /etc/tripwire/twpol.txt
```

zu aktuallisieren. Dadurch wird auch die alte Datenbank an die neuen Richtlinien angepasst, wobei alte Daten weiterverwendet werden, lediglich um neu erforderliche aufgestockt. Eine Manipulation der Dateien wird damit auch bei einer Änderung an den Richtlinien entdeckt. Weil dieses Kommando sowohl Richtlinien als auch Datenbank verändert, müssen Sie sowohl den lokalen als auch den globalen Schlüssel angeben.


Für den Anfang ist es sinnvoll, mit einer möglichst vollständigen Überwachung zu starten und dann nach und nach die Überwachungsrichtlinien dort einzuschränken, wo falsch-positive Meldungen auftreten. Lediglich die Inhalte der Verzeichnisse `/home/*`, `/tmp`, `/var/tmp` und einiger anderer (aber nicht die Verzeichnisse selbst!) sollten Sie gleich von Anfang an herauslassen, da sie nicht überwachbar sind.

Übungen

7.4 [2] Diskutieren Sie die Vor- und Nachteile von signierten Datenbanken gegenüber Datenbanken auf schreibgeschützten Medien.

7.5 [2] Angenommen, Sie haben die Tripwire-Datenbank auf einem schreibgeschützten Medium platziert. Welche Möglichkeiten hat ein Angreifer dennoch, um einer Entdeckung durch Tripwire zu entgehen.

7.6 [2] Legen Sie ein Verzeichnis an und bevölkern Sie es mit einigen Dateien:

```
# mkdir /tmp/tw-test
# cp /bin/rm /bin/ls /bin/ps /tmp/tw-test
```

Erstellen Sie nun wie in Abschnitt 7.2.3 beschrieben eine Richtliniendatei mit der Richtlinie

```
/tmp/tw-test  ->          $(ReadOnly);
```

und erstellen Sie die Referenz-Datenbank und einen ersten Bericht. Entfernen Sie eine der Dateien in `/tmp/tw-test`, ändern Sie den Eigentümer der zweiten und verändern Sie den Inhalt der dritten.

Erstellen Sie einen Bericht und schauen Sie sich auch die Langform des Berichts durch `twprint` an.

7.7 [2] Legen Sie zwei inhaltsgleiche Dateien an sowie eine symbolische Verknüpfung auf eine dieser Dateien:

```
$ date > datei-a
$ cp datei-a datei-b
$ ln -s datei-a link
```

Erstellen Sie nun eine Tripwire-Datenbank, die auch die symbolische Verknüpfung `link` enthält. Welche Tests von Tripwire springen an, wenn Sie die Verknüpfung nun durch

```
$ ln -sf datei-b link
```

auf `datei-b` verbiegen?

7.8 [1] Warum ist eine Überwachungsrichtlinie wie

```
/bin          ->          $(IgnoreNone);
```

nicht sinnvoll? Erstellen Sie ggf. eine solche Überwachungsrichtlinie und testen Sie das Verhalten von Tripwire durch mehrfache Berichtserstellung.

7.9 [2] Warum ist es nicht sinnvoll, die Tests `a` (Zugriffszeit) oder `m` (Änderungszeit) ohne den Test `c` (Metadaten-Änderungszeit) durchzuführen?

7.10 [2] Die vordefinierte Variable `Device` enthält eine Test-Auswahl, die u. a. für Gerätedateien gedacht ist. Allerdings ist diese Auswahl nicht immer geeignet, weil einige Gerätedateien den Besitzer und die Zugriffsrechte wechseln können. Starten Sie mit der Richtlinie

```
/dev      ->      $(Device);
```

und finden Sie heraus, welche Gerätedateien das betrifft.

7.11 [2] Welche Dateien in `/etc` ändern sich im Normalbetrieb? Beantworten Sie diese Frage mit Hilfe Ihres Linux-Wissens und indem Sie es mit Tripwire überprüfen.

7.12 [2] In welchen Verzeichnissen müssen Sie im Normalbetrieb mit dem Anlegen *neuer* Dateien rechnen? Beantworten Sie diese Frage mit Hilfe Ihres Linux-Wissens und indem Sie es mit Tripwire überprüfen.

7.3 AIDE

7.3.1 Einleitung

Ein Bruder im Geiste des im vorigen Abschnitt vorgestellten Programms `tripwire` ist das *Advanced Intrusion Detection System*, kurz AIDE [aide]. Auch mit diesem von Rami Lehti, Pablo Virolainen und Richard van den Berg entwickelten Programm werden Dateien anhand von Schlüsselattributen und Hashwerten mit einer Datenbank verglichen und Änderungen aufgezeigt. Es wurden jedoch einige andere Hashalgorithmen eingebaut und die Konfigurationsmöglichkeiten im Vergleich zu Tripwire erweitert.

Unterschied zu Tripwire

7.3.2 Arbeitsmodi von AIDE

Die verschiedenen Arbeitsmodi von AIDE werden per Kommandozeilenparameter festgelegt.

`--check` Prüft die Integrität der zu überwachenden Dateien mit Hilfe einer vorher angelegten Vergleichsdatenbank, üblicherweise `/etc/aide.db`. Dieser Arbeitsmodus ist auch die Voreinstellung von AIDE, die Angabe von `--check` kann daher entfallen.

`--init` Erzeugt die Vergleichsdatenbank standardmäßig als `/etc/aide.db.new`. Diese muss vor der Verwendung mit `--check` noch an den richtigen Ort im Dateibaum transferiert bzw. umbenannt werden.

`--update` Aktualisiert die Vergleichsdatenbank nichtinteraktiv und schreibt eine neue Datenbankdatei.

`--compare` Vergleicht zwei Datenbanken, die in der Konfigurationsdatei mit den Angaben `database=` und `database_new=` definiert werden müssen.

Außerdem unterstützt AIDE auch einige andere Kommandozeilenschalter, etwa diese:

`--config=`⟨*conffile*⟩ Liest die Konfiguration aus der angegebenen Datei statt aus `/etc/aide.conf`. Die Angabe eines Minuszeichens erlaubt es, die Konfiguration über `stdin` einzulesen.

`--verbose=`⟨*level*⟩ Verändert die Geschwätzigkeit des Programms. Vorgabewert ist 5, bei der Angabe von `--verbose` ohne Parameter wird der Wert auf 20 gesetzt. Maximale Informationsausgabe wird mit 255 errreicht.

`--report=`⟨*URL*⟩ Weist AIDE an, seine Ausgaben an den angegebenen URL zu schicken.

7.3.3 Konfiguration von AIDE

Die Konfiguration von AIDE wird normalerweise in der Datei `/etc/aide.conf` vorgenommen. Die Syntax ähnelt sehr der von `tripwire`, so dass eine `tw.config`-Datei leicht in eine AIDE-Konfiguration überführt werden kann. Die Eintragungen lassen sich in drei Typen einteilen. Zunächst finden sich Konfigurationseinträge, die Variablen setzen und allgemeine Einstellungen beschreiben. Die Selektionszeilen dienen zur Festlegung der zu überwachenden Dateien und der gewünschten Testverfahren. Schließlich können noch Makrozeilen angegeben werden.

`/etc/aide.conf`

Tabelle 7.2: Dateiattribute für AIDE

Attribut	Bedeutung
`p`	Dateirechte (engl. *permission*)
`i`	Inode-Nummer (engl. *inode*)
`n`	Referenzzähler (engl. *number of links*)
`u`	Dateibesitzer (engl. *user*)
`g`	Dateigruppe (engl. *group*)
`s`	Dateigröße (engl. *size*)
`a`	Zugriffzeit (engl. *access*)
`m`	Inhaltsänderungszeit (engl. *modification*)
`c`	Inodeänderungszeit (engl. *change*)
`S`	zunehmende Dateigröße (engl. *growing size*)

Konfigurationszeilen Diese Einträge sind nach dem Schema »⟨*Parameter*⟩=⟨*Wert*⟩« aufgebaut. Einige wichtige Angaben sind etwa:

`database=`⟨*URL*⟩ Dort findet sich die Vergleichsdatenbank; Vorgabe ist `/etc/aide.db`.

`database_out=`⟨*URL*⟩ An dieser Stelle landen neu erstellte Datenbankdateien. Vorgabe ist `/etc/aide.db.new`.

`database_new=`⟨*URL*⟩ Legt fest, wo sich die Vergleichsdatenbank bei der Verwendung von `--compare` befindet.

`report_url=`⟨*URL*⟩ Gibt an, wohin AIDE seine Berichte ausliefern soll; Vorgabe ist `stdout`.

Selektionszeilen Wiederum lassen sich hier drei Varianten unterscheiden. Zeilen, die mit einem `/` beginnen, sind normale Auswahlzeilen. Steht als erstes Zeichen in der Zeile ein `!`, wird dies als Ausschlusseintrag aufgefasst. Ein `=` am Zeilenbeginn erfordert exakte Übereinstimmung mit der Angabe.

Makrozeilen Hier ist es möglich, Variable und bedingte Verzweigungen einzurichten. Genauere Informationen finden sich in der Dokumentation der Konfigurationsdatei (`aide.conf`(5)).

URL-Angaben Quellen und Ziele für Ein- und Ausgaben von AIDE lassen sich mit Hilfe von URLs festschreiben. Hier sind folgende Varianten verfügbar:

`stdout` Schreibt auf die Standardausgabe.

`stderr` Schreibt auch auf den Standardfehlerkanal.

`stdin` Liest vom Standardeingabekanal.

`file://datei` Liest aus bzw. schreibt in die angegebene Datei.

`fd:`*n* Liest aus bzw. schreibt in den angegebenen Filedeskriptor.

Auswahlregeln AIDE überwacht einen ähnlichen Satz von Dateiattributen wie Tripwire (Tabelle 7.2). Die folgenden Prüfsummenverfahren werden unterstützt:

Prüfsummenverfahren

`md5` Das MD5-Verfahren von Ron Rivest

`sha1` Der verbesserte *Secure Hash Algorithm* des NIST

`rmd160` RMD160 wurde im Rahmen des europäischen RIPE-Projektes von Hans Dobbertin, Anton Bosselaers und Bart Preneel entwickelt. Der Algorithmus *RIPE Message Digest* mit Hashwerten von 160 Bit Länge gilt als sehr sicher.

tiger Dieser Algorithmus wurde von Eli Biham und Ross Anderson entwickelt und kann Hashwerte von bis zu 192 Bit Länge erzeugen.

Mit aktiverter »mhash«-Unterstützung sind weiterhin verfügbar:

crc32 Die 32-Bit-Variante des *Cyclic Redundancy Check*

haval Der Haval-Algorithmus von Yuliang Zheng

gost GOST ist ein Hashalgorithmus mit 256-Bit-Prüfsumme, der in der damaligen Sowjetunion entwickelt wurde.

Vorgegebene Kombinationen der Auswahlkriterien sind diese:

R entspricht `p+i+n+u+g+s+m+c+md5`

L ist eine Kombination aus `p+i+n+u+g`

E steht für *empty*, also eine leere Auswahl

> dient zur Überwachung stetig wachsender Dateien mit `p+u+g+i+n+S`

7.3.4 Beispielkonfiguration von AIDE

Im Installationsumfang von AIDE befindet sich eine gut dokumentierte Beispielkonfiguration, die leicht an eigene Systeme angepasst werden kann (Bild 7.1).

Übungen

7.13 [2] Führen Sie die Übungen zu Tripwire sinngemäß mit AIDE durch.

```
# AIDE conf

database=file:/var/lib/aide/aide.db
database_out=file:/var/lib/aide/aide.db.new

# Change this to "no" or remove it to not gzip output
# (only useful on systems with few CPU cycles to spare)
gzip_dbout=yes

# Here are all the things we can check - these are the default rules
#
#p:      permissions
#i:      inode
#n:      number of links
#u:      user
#g:      group
#s:      size
#b:      block count
#m:      mtime
#a:      atime
#c:      ctime
#S:      check for growing size
#md5:    md5 checksum
#sha1:   sha1 checksum
#rmd160: rmd160 checksum
#tiger:  tiger checksum
#R:      p+i+n+u+g+s+m+c+md5
#L:      p+i+n+u+g
#E:      Empty group
#>:      Growing logfile p+u+g+i+n+S
#haval:         haval checksum
#gost:          gost checksum
#crc32:         crc32 checksum

# Defines formerly set here have been moved to /etc/default/aide.

# Custom rules
Binlib = p+i+n+u+g+s+b+m+c+md5+sha1
ConfFiles = p+i+n+u+g+s+b+m+c+md5+sha1
Logs = p+i+n+u+g+S
Devices = p+i+n+u+g+s+b+c+md5+sha1
Databases = p+n+u+g
StaticDir = p+i+n+u+g
ManPages = p+i+n+u+g+s+b+m+c+md5+sha1
```

Bild 7.1: Beispielkonfiguration für AIDE (Teil 1)

```
# Next decide what directories/files you want in the database

# Kernel, system map, etc.
=/boot$ Binlib
# Binaries
/bin Binlib
/sbin Binlib
/usr/bin Binlib
/usr/sbin Binlib
/usr/local/bin Binlib
/usr/local/sbin Binlib
/usr/games Binlib
# Libraries
/lib Binlib
/usr/lib Binlib
/usr/local/lib Binlib
# Log files
=/var/log$ StaticDir
!/var/log/ksymoops
/var/log/aide/aide.log(.[0-9])?(.gz)? Databases
/var/log/aide/error.log(.[0-9])?(.gz)? Databases
/var/log/setuid.changes(.[0-9])?(.gz)? Databases
!/var/log/aide
/var/log Logs
# Devices
!/dev/pts
# If you get spurious warnings about being unable to mmap()
# /dev/cpu/mtrr, you may uncomment this to get rid of them. They're
# harmless but sometimes annoying.
#!/dev/cpu/mtrr
!/dev/xconsole
/dev Devices
# Other miscellaneous files
/var/run$ StaticDir
!/var/run
# Test only the directory when dealing with /proc
/proc$ StaticDir
!/proc

# You can look through these examples to get further ideas

# MD5 sum files - especially useful with debsums -g
#/var/lib/dpkg/info/([^\.]+).md5sums u+g+s+m+md5+sha1
```

Bild 7.2: Beispielkonfiguration für AIDE (Teil 2)

```
# Check crontabs
#/var/spool/anacron/cron.daily Databases
#/var/spool/anacron/cron.monthly Databases
#/var/spool/anacron/cron.weekly Databases
#/var/spool/cron Databases
#/var/spool/cron/crontabs Databases
# manpages can be trojaned, especially depending on *roff implementation
#/usr/man ManPages
#/usr/share/man ManPages
#/usr/local/man ManPages

# docs
#/usr/doc ManPages
#/usr/share/doc ManPages

# check users' home directories
#/home Binlib

# check sources for modifications
#/usr/src L
#/usr/local/src L

# Check headers for same
#/usr/include L
#/usr/local/include L
```

Bild 7.3: Beispielkonfiguration für AIDE (Teil 3)

Kommandos in diesem Kapitel

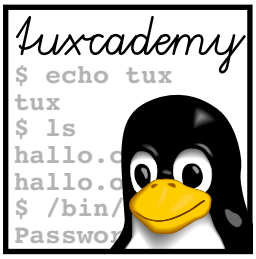
tripwire	Vergleicht Datei-Prüfsummen mit einer Datenbank	tripwire(8)	108
twadmin	Verwaltungsprogramm für Tripwire	twadmin(8)	107
twprint	Gibt Tripwire-Datenbank aus	twprint(8)	108

Zusammenfassung

- Rechnerbasierte Angriffserkennung soll Manipulationen an Programmen und wichtigen Systemdateien auf einem Rechner aufdecken.
- Abweichungen können durch eine Datenbank des Soll-Zustandes oder durch Signaturen von Schädlingen erkannt werden.
- Tripwire und AIDE sind zwei ähnliche Systeme, die systematisch kryptographische Prüfsummen für die Dateien eines Rechners aufstellen und spätere Vergleiche ermöglichen.

Literaturverzeichnis

aide »AIDE – Advanced Intrusion Detection Environment«.
http://www.cs.tut.fi/~rammer/aide.html



8

Netzbasierte Angriffserkennung

Inhalt

Lernziele

- Techniken zur netzbasierten Angriffserkennung kennen
- Port-Scans mit dem `scanlogd` erkennen können
- Mit `fail2ban` plumpe Kennwort-Rateangriffe auf Dienste wie SSH unterbinden können
- Die Grundlagen von Snort kennen

Vorkenntnisse

- Kenntnisse der Linux-System- und -Netzadministration
- TCP/IP-Kenntnisse

8.1 Einleitung

Wenn ein Rechner im Netzwerk erreichbar ist, heißt dies, dass er auch verwundbar ist. Dies trifft sowohl auf die Arbeitsplätze von Benutzern, aber in besonderem Maße auch auf Server zu, die bestimmte Dienste innerhalb des internen Netzwerkes oder gar weltweit anbieten. Diese Dienste stehen allgemein zur Verfügung und sind daher auch ein naheliegendes Ziel für Angriffsversuche.

Der Zugang zu solchen Maschinen kann über Firewalls genau reglementiert werden, was bereits eine bedeutende Erhöhung der Sicherheit darstellt. Aber auch Firewalls selbst sind angreifbar oder können bei unzureichender Konfiguration umgangen werden, daher sollten möglichst alle Rechner mit geeigneten Maßnahmen gegen Einbruchs- und Manipulationsversuche abgesichert werden. Diese können zum einen rechnerbasiert versuchen, Manipulationen an wichtigen Programmen und Dateien aufzudecken (Kapitel 7); die Klasse der »netzbasierten Angriffserkennungssysteme« versucht dagegen, den Datenverkehr auf dem Netz zu analysieren, daraus abzuleiten, ob ein Angriff im Gange ist und wenn ja, dessen Natur und Ursprung einzugrenzen.

Erkundung des Netzes

Die übliche Vorgehensweise, die ein potenzieller Angreifer an den Tag legen wird, ist zunächst einmal die Erkundung des Netzes. Wenn mit Hilfe geeigneter Programme erst einmal die Struktur des Netzes bekannt ist sowie eine Liste der angebotenen Dienste vorliegt, ist es nur noch ein kleiner Schritt, die passenden Angriffsprogramme (engl. *exploits*) zu starten und so bestimmte Maschinen im Netz auszuschalten oder zu manipulieren. Eine wichtige Programmklasse zur Erkundung fremder Netzwerke sind Portscanner wie etwa das Programm `nmap`.

Portscanner

Zur Einrichtung einer »virtuellen Einbruchsalarmanlage«, wie man Angriffserkennungssysteme auch nennen könnte, stehen unter Linux verschiedene Werkzeuge zur Verfügung.

Derartige Einrichtungen ermöglichen die Erkennung und möglicherweise Abwehr von Angriffsversuchen. Neben der Rückverfolgung des Angreifers werden im Idealfall auch Informationen geliefert, wie ein ggf. angerichteter Schaden wieder behoben werden kann.

Grenzen

Automatische Angriffserkennungssysteme haben aber auch ihre Grenzen. Bruce Schneier [Sch04] gibt hierzu ein anschauliches Beispiel: Nehmen wir an, für eine seltene Krankheit existiert ein Test mit 99% Genauigkeit – von 100 Gesunden wird eine Person irrtümlich mit der Krankheit diagnostiziert, und von 100 Kranken wird eine Person irrtümlich für gesund erklärt. Wenn in der Bevölkerung eine Person von 10.000 tatsächlich von der Krankheit befallen ist, ist der Test wertlos, da nur 1% der Personen, bei denen der Test ein positives Ergebnis liefert, wirklich an der Krankheit leidet. Das Problem sind die »falschen Positiven«, die so zahlreich auftreten, dass die »echten Positiven« quasi im Rauschen verschwinden. Viele Angriffserkennungssysteme haben dasselbe Problem, übermäßige Fehlalarme führen aber zur Abstumpfung seitens der Administratoren und möglicherweise dazu, dass auf einen der (seltenen) echten Angriffe zu spät oder gar nicht reagiert wird.

Falsche Positive

Sie sollten sich immer vor Augen führen, dass ein Angriffserkennungssystem – wie eine Alarmanlage – keinen Bestandteil der Infrastruktur zur Abwehr von Angriffen darstellt. Das Angriffserkennungssystem kommt erst dann zum Tragen, wenn die Abwehrinfrastruktur (Paketfilter, Proxies, …) bereits *versagt* hat. Jegliche Abwehrmaßnahmen, die das Angriffserkennungssystem einleitet, etwa eine Umkonfiguration des Paketfilters, um Datagramme des oder der angreifenden Rechner(s) abzuweisen, tritt erst ein, wenn möglicherweise bereits Schaden angerichtet wurde.

8.2 Portscans erkennen – scanlogd

Wie bereits erläutert ist der erste Schritt eines Einbruchsversuches üblicherweise die Analyse des Netzwerkes durch einen Portscanner. Zu diesem Zweck werden vom Angreifer in meist rascher Folge Verbindungen zu allen möglichen und unmöglichen Ports aufgebaut.

Portscanner

Falls das Zielnetzwerk durch einen Linux-Firewall abgesichert ist, können mit Hilfe der im Paketfiltermechanismus des Kernels eingebauten Protokollfunktion Zugriffsversuche auf gesperrte Ports festgehalten werden. In der zentralen Protokolldatei `/var/log/messages` werden unter anderem Datum, Uhrzeit, IP-Adresse des Absenders und die Schnittstelle, über die das Paket empfangen wurde, festgehalten. Mit einem geeigneten Skript, das die Protokolldatei in regelmäßigen Zeitintervallen durchsucht, lassen sich also ungewöhnliche Häufungen solcher Kontaktversuche ermitteln und eine entsprechende Meldung an den Systemadministrator absenden.

Linux-Firewall

Zugriffsversuche auf gesperrte Ports

Dem Vorteil einer zentralen Überwachung von Portscans an der Firewall steht der Nachteil gegenüber, dass nur von außerhalb initiierte Angriffsversuche ermittelt werden können. Statistiken zeigen jedoch, dass etwa vier Fünftel aller Attacken von innen durchgeführt werden. Bei solchen Varianten muss diese Lösung zwangsläufig scheitern.

Attacken von innen

Um nun aber Server auch gegen interne Portscans absichern zu können, ohne auf jeder Maschine gleich eine Paketfilterfunktionalität einrichten zu müssen, existiert unter Linux ein kleines Programm namens `scanlogd`. Dieser Dienst gehört bei vielen Distributionen zum Lieferumfang und bietet zwar nicht so mächtige Funktionen wie andere Programme zur Angriffserkennung, dafür kann `scanlogd` aber ohne umfangreiche Konfigurationsarbeit rasch eingerichtet werden.

`scanlogd`

Versprechen Sie sich nicht zuviel von der automatischen Portscan-Erkennung. `scanlogd` funktioniert recht zuverlässig, aber ein Portscan ist heute nichts Ungewöhnliches mehr – Sie können damit rechnen, dass ein frisch ans Internet angeschlossener Rechner nach längstens einer Viertelstunde oder so Ziel eines Portscans wird. Bevor Sie also bei jedem Portscan die Alarmglocken klingeln lassen, sollten Sie beobachten, wie oft so etwas bei Ihnen wirklich vorkommt und in wie vielen Fällen sich ein gefährlicher Angriff anschließt; es könnte sein, dass Ihr `scanlogd` die Rolle des Hirtenjungen in der Fabel einnimmt, der zu oft »Wolf!« gerufen hatte und dem dann niemand mehr glauben wollte, als tatsächlich ein Wolf vorbeikam …

Der `scanlogd` protokolliert erkannte Portscans per `syslogd`. Sinnvollerweise sollte der Dienst also durch ein Init-Skript in den Runlevels mit Netzwerkunterstützung nach der Aktivierung von Netzwerkkarte und `syslogd` automatisch gestartet werden.

`syslogd`

Bei seinem Start besetzt der `scanlogd` ein Socket und überwacht darüber alle durch den Rechner empfangenen IP-Datagramme. Sendet nun eine Station in kurzer Zeit viele Datagramme an verschiedene Ports des Rechners, wird dieses Ereignis als Portscan interpretiert und festgehalten. Um Fehlalarme zu vermeiden, sieht der `scanlogd` erst dann Verbindungsversuche als »Scan« an, wenn mindestens 7 verschiedene privilegierte oder 21 nicht privilegierte Ports im Abstand von höchstens drei Sekunden durch denselben Absender angesprochen werden. Um ein solches Ereignis zu protokollieren, müssen nun mehr als 5 Scans innerhalb von 20 Sekunden erfolgen.

Funktionsweise

Die Meldungen werden an den `syslogd` für die Kategorie `daemon.alert` mit der Identifikation `scanlogd` weitergeleitet und landen meistens in der zentralen Protokolldatei `/var/log/messages`. Dort finden sich dann einzelne Zeilen mit dem allgemeinen Aufbau

Meldungen

⟨*QAdr*⟩ `to` ⟨*ZAdr*⟩ `ports` ⟨*ports*⟩`,` `flags` ⟨*tcp-flags*⟩`,`▷
◁ `TOS` ⟨*bits*⟩`,` `TTL` ⟨*ttl*⟩`,` `@`⟨*hh:mm:ss*⟩

Hier ist eine solche Meldung exemplarisch zu sehen:

```
# tail /var/log/messages
◁◁◁◁◁
Aug  8 14:36:32 rechner scanlogd: 192.168.0.99 to 192.168.0.100 ▷
 ◁ ports 1542, 438, 1370, 7009, 7006, 604, 32773, ..., fSrpauxy,▷
 ◁ TOS 00, TTL 64 @14:36:32
◁◁◁◁◁
```

Zur Überwachung des Systems sollten die Meldungen des `scanlogd` mit einem geeigneten Programm aus der Datei `/var/log/messages` ausgelesen werden.

Übungen

8.1 [!3] Installieren Sie `scanlogd` und versuchen Sie, mit einem Programm wie `nmap` einen Portscan durchzuführen. Wird dieser von `scanlogd` gemeldet? Können Sie mit `nmap` einen Portscan durchführen, den `scanlogd` nicht protokolliert?

8.3 Angreifer aussperren – `fail2ban`

8.3.1 Überblick

Ärgerlicher als Portscans sind Versuche, beispielsweise SSH-Zugänge auf schwache Kennwörter abzuklopfen. Das ist normalerweise mit dumpfer Probiererei verbunden, aber – im Gegensatz zum Zahlenschloss am Fahrrad oder Koffer, wo es zwar nur 1000 Möglichkeiten gibt, man sich jedoch in unmittelbarer physikalischer Nähe zum Objekt des Begehrens befinden muss – ist es über das Internet relativ flott möglich, ziemlich viele Kandidaten-Kennwörter zu testen. Als Systemadministrator sollten Sie daher aufpassen. Das Programm `fail2ban` liest das Systemprotokoll und versucht, solche plumpen Crack-Versuche zu erkennen. Findet es einen, so kann es die IP-Adresse des betreffenden Rechners über `iptables` oder den TCP-Wrapper sperren und damit weitere Versuche unterbinden. Nach einer Weile wird die Adresse wieder freigegeben.

Grundsätzlich können Sie das Problem an der Wurzel ausrotten, indem Sie Ihre SSH-Zugänge nicht über Kennwörter, sondern über asymmetrische Kryptografie schützen (die entsprechenden Schlüssel sind praktisch unmöglich zu erraten). Selbst dann kann es aber noch nützlich sein, `fail2ban` laufen zu lassen, damit die Skript-Kiddies Ihnen nicht die Netzanbindung und die Protokolldateien zumüllen.

Außer der SSH stehen auch andere kennwortgeschützte Dienste wie POP3, IMAP oder FTP auf der Liste von Angreifern. `fail2ban` kann sich um die allermeisten davon kümmern.

8.3.2 Struktur

Das Programm `fail2ban` besteht aus einem Client und einem (als Daemon laufenden) Server, `fail2ban-client` und `fail2ban-server`. Der Server wird indirekt vom Client aus gestartet und bekommt vom Client auch seine Konfiguration übertragen. Anschließend übernimmt er die eigentliche Arbeit, nämlich das Beobachten von Protokolldateien und die Interaktion mit dem Sperrmechanismus (typischerweise `iptables`).

Jails
Filter
Aktion

Der `fail2ban-server` kann gleichzeitig mehrere »Jails« verwalten. Ein Jail ist eine Kombination aus einem »Filter« und einer oder mehrerer »Aktionen«. Ein Filter enthält reguläre Ausdrücke, die in Protokolldateien gesucht werden. Eine Aktion

bestimmt Kommandos, die zu verschiedenen Gelegenheiten ausgeführt werden, etwa um eine Adresse zu sperren oder zu entsperren.

Die Jails für den Server werden in der Datei /etc/fail2ban/jail.conf definiert. Dort könnte zum Beispiel etwas stehen wie

```
[DEFAULT]
ignoreip = 127.0.0.1                          Ignoriere diesen Rechner
bantime = 600                                 Sperre Übeltäter für 10 Minuten
maxretry = 3                                  Dulde maximal drei Fehltritte
backend = polling                             Methode zum Lesen der Protokolldateien
destemail = root@localhost                    Wird in Aktionen benutzt
banaction = iptables-multiport                Standard-Aktion
mta = sendmail                                Verschickt Mail à la Sendmail (statt mail)
protocol = tcp
```

Dies sind Vorgaben, die für alle Jails gelten (und in den Definitionen der einzelnen Jails überschrieben werden können). Eine typische Jail-Definition könnte aussehen wie

Jail-Definition

```
[ssh]                                         Name des Jails
enabled = true                                Aktiv geschaltet
port = ssh                                    Überwache Port 22 (aus /etc/services)
filter = sshd                                 Verwende Filter aus /etc/fail2ban/filter.d/sshd.conf
logpath = /var/log/auth.log                   Betrachte diese Protokolldatei
maxretry = 6                                  Beim siebten Fehlversuch fällt der Hammer
```

Die Filterdefinitionen finden sich im Verzeichnis /etc/fail2ban/filter.d. Hier ist ein Auszug aus der Datei sshd.conf:

Filterdefinitionen

```
[INCLUDES]
before = common.conf                          Definitionen für __prefix_line usw.

[Definition]
_daemon = sshd

failregex = ^%(__prefix_line)s(?:error: PAM: )?Authentication▷
                          ◁ failure for .* from <HOST>\s*$
    ^%(__prefix_line)sFailed (?:password|publickey) for .* ▷
             ◁ from <HOST>(?: port \d*)?(?: ssh\d*)?$
    ^%(__prefix_line)sROOT LOGIN REFUSED.* FROM <HOST>\s*$
    ^%(__prefix_line)s[iI](?:llegal|nvalid) user .* ▷
                                        ◁ from <HOST>\s*$
    ^%(__prefix_line)sUser .+ from <HOST> not allowed ▷
                             ◁ because not listed in AllowUsers$
    ^%(__prefix_line)srefused connect from ▷
                  ◁ \S+ \(<HOST>\)\s*$
    ^%(__prefix_line)sAddress <HOST> .* POSSIBLE BREAK-IN ▷
                  ◁ ATTEMPT!*\s*$

ignoreregex =
```

(einige überlange Zeilen wurden aus Gründen der Übersicht unterschlagen). failregex enthält eine Folge von regulären Ausdrücken, die in den mit logpath in der Jail-Definition angegebenen Protokolldateien gesucht werden. Jeder Treffer für einen bestimmten entfernten Rechner erhöht den Zähler um 1; wird der Wert von maxretry aus der Jail-Definition überschritten, sperrt fail2ban den betreffenden Rechner.

`<HOST>` in den regulären Ausdrücken ist eine bequeme Abkürzung für `(?:::f{4,6}:)?(?P<host>[\w\-.^_]+)`, also im Wesentlichen einen Rechnernamen oder eine IP-Adresse.

Sie können Zeilen in Protokolldateien völlig ignorieren, indem Sie passende reguläre Ausdrücke in `ignoreregex` angeben.

Übungen

8.2 [!2] Überzeugen Sie sich, dass `fail2ban` funktioniert: Installieren Sie das Programm und stellen Sie sicher, dass das `ssh`-Jail aktiv ist. Verwenden Sie dann einen anderen (virtuellen?) Rechner oder bitten Sie im Präsenztraining Ihren Sitznachbarn, sich einige Male per `ssh` mit einem beliebigen Kennwort mit Ihrem Rechner zu verbinden zu versuchen. Beobachten Sie die Protokolldateien von `sshd` und `fail2ban` auf Ihrem Rechner. Wird der entfernte Rechner gesperrt?

8.3 [2] (Fortsetzung von Übung 8.2) Verwenden Sie `iptables`, um nachzuschauen, wie die Sperrung des entfernten Rechners technisch realisiert wird. Studieren Sie hierzu auch die im Jail angesprochene Aktions-Definition (normalerweise unter `/etc/fail2ban/action.d` zu finden).

8.4 [1] (Fortsetzung von Übung 8.2) Vergewissern Sie sich auch, dass der entfernte Rechner nach `bantime` Sekunden wieder entsperrt wird.

8.4 Snort: Schweinereien in Echtzeit erkennen

8.4.1 Grundlagen

Die Erkennung von Portscans und Angriffen auf einzelne Dienste ist sicher nützlich und hilfreich. Aber die Software dafür läuft immer noch auf jedem einzelnen Rechner und sichert auch nur diesen. Wie wäre es eigentlich, wenn man den kompletten Netzverkehr anschauen und auf problematische Aktivitäten abklopfen könnte? Das ist die Domäne von netzbasierten Angriffserkennungssystemen (IDS) wie »Snort«. Sie suchen im Netzverkehr nach verdächtigen Mustern von Zugriffen und schlagen gegebenenfalls Alarm. Auf den ersten Blick klingt sowas natürlich extrem nützlich, aber bevor wir uns Snort etwas genauer anschauen, müssen wir noch ein paar grundlegende Fragen ansprechen.

Für die LPI-202-Prüfung müssen Sie wissen, dass Snort existiert und was es in etwa tut. Es dürfte also reichen, wenn Sie diesen Abschnitt (bis zum Ende des Kapitels) nur überfliegen.

Wo sollte man mitlesen? Es gibt im Wesentlichen drei Bereiche, wo Sie ein netzbasiertes IDS in Stellung bringen können:

An der Außenanbindung Das ist möglicherweise naheliegend, aber relativ zwecklos. Zum einen setzen Sie sich damit jeder Menge falscher Positive aus, und zum anderen sollte Ihr äußerer Paketfilter den allergrößten Teil der verdächtig aussehenden Zugriffe abwehren. Selbst bei echten Angriffen lautet die sinnvolle Reaktion daher in den allermeisten Fällen »Ignorieren«.

In der DMZ Das ergibt schon eher Sinn. Ein netzbasiertes IDS an dieser Stelle erlaubt Ihnen Schlüsse darüber, wie zuverlässig Ihr äußerer Paketfilter seine Sicherheitsvorgaben umsetzt, und alarmiert Sie möglicherweise über unerwünschte Aktivitäten auf dem Netz der DMZ, die zu weiteren Angriffen führen könnten.

Im internen Netz Wenn Sie hier einen Angriff von außen feststellen, haben Sie offensichtlich ein kapitales Problem, von dem Sie wissen sollten. Außerdem könnte es sein, dass Sie es mit internen Angreifern zu tun bekommen, für die Ihr Firewallsystem kein Hindernis darstellt.

Wie sollte man mitlesen? Das wichtigste Ziel der Angriffserkennung ist, selber unerkannt zu bleiben. Nur auf diese Weise läßt sich sicherstellen, dass das IDS nicht selbst zum Ziel der Eindringlinge wird, die versuchen, unbemerkt zu bleiben, indem sie das IDS außer Gefecht setzen.

Idealerweise läuft das IDS auf einem völlig separaten Rechner (und nicht auf einem Rechner, der für Cracker verlockend aussieht, etwa weil er einen anderen wichtigen und kompromittierbaren Dienst wie Web, Mail oder DNS erbringt).

Das Problem besteht in diesem Fall darin, dafür zu sorgen, dass das IDS tatsächlich den kompletten zu untersuchenden Datenverkehr zu sehen bekommt. Heutige Switches vermitteln Daten ja direkt nur zwischen den beteiligten Anschlüssen, statt wie in der guten alten Ethernet-Zeit alle Daten aufs »lange Kabel« zu schicken, wo alle Stationen sie lesen können. Gute Switches erlauben es aber meistens, zumindest einen Anschluss in einen »Managementmodus« zu versetzen, der parallel zu den eigentlich beteiligten Anschlüssen den kompletten über den Switch laufenden Datenverkehr geschickt bekommt.

Wobei sich schon das nächste Problem stellt: Die interne Vermittlungskapazität guter Switches liegt um ein Mehrfaches über der Datenrate, die an einzelnen Anschlüssen zur Verfügung steht (Sie möchten schließlich, dass sich die Stationen A und B sowie die Stationen X und Y parallel mit der vollen Datenrate ihrer Anschlüsse unterhalten können). Das heißt aber, dass ein einzelner Anschluss im Managementmodus nicht den kompletten vom Switch vermittelten Datenverkehr transportieren kann. – Zum Glück ist das meist nur ein Problem, wenn Sie im internen Netz lauschen wollen; in der DMZ wird die Menge der »interessanten« Daten normalerweise durch die Geschwindigkeit der Außenanbindung limitiert.

Für 10-Mbit/s- und 100-Mbit/s-Ethernet ist es mit relativ geringem Aufwand möglich, »passive Taps« zu bauen, die man in ein Netzwerkkabel einschleifen kann, um den kompletten Verkehr, der über das Kabel läuft, an einen anderen Rechner weiterzuleiten (dieser Rechner braucht zwei Netzwerkschnittstellen, um den ganzen Verkehr sehen zu können, da im Vollduplexbetrieb zwei Aderpaare im Kabel parallel für die Übertragung benutzt werden, eine einzige Netzwerkschnittstelle aber nur ein Aderpaar lesen kann). Suchen Sie mit der Suchmaschine Ihres Vertrauens nach »*passive ethernet tap*«.

Bei Gigabit-Ethernet können Sie keine passiven Taps bauen, weil die Übertragungsverfahren dafür zu kompliziert sind. Abhilfe schafft ein »aktiver Tap« in Gestalt eines Linux-Rechners mit zwei Gigabit-Netzwerkschnittstellen und `ebtables`.

Der Vorteil eines »passiven Taps« ist, dass der IDS-Rechner aus dem überwachten Netz überhaupt nicht zu sehen ist, weil er keine Möglichkeit hat, elektrische Signale auf das Kabel zu schicken. Ein Angreifer kann also nicht wissen, dass seine Aktivitäten überwacht werden. Der IDS-Rechner kann bzw. sollte natürlich über ein separates »Überwachungsnetz« erreichbar sein, das nicht mit dem überwachten Netz verbunden ist, damit er Alarmmeldungen und ähnliches weiterleiten kann.

Im Zeitalter der Virtualisierung können Sie natürlich eine »virtuelle DMZ« auf einem einzigen physikalischen Rechner haben und die IDS-Infrastruktur ebenfalls auf diesem Rechner unterbringen. Das ist nicht ganz so sicher, aber weit weniger aufwendig.

Was ist von einem netzbasierten IDS zu erwarten? Die Aufgabe eines netzbasierten IDS ist, Anomalien im Netzverkehr zu erkennen, die auf Angriffe hindeuten. Dafür können einerseits Regelsätze verwendet werden, die verdächtige Aktivitäten abstrakt beschreiben; andererseits kann das IDS aber auch auf »Fingerabdrücke« bekannter Angriffe zurückgreifen, um gängige Schadsoftware erkennen zu können. Dabei sind auch Eigenheiten der Datenübertragung per TCP/IP zu berücksichtigen.

Historisch haben netzbasierte IDS einzelne Datagramme betrachtet. Zum Beispiel könnte ein TCP-Segment, in dem gleichzeitig das `SYN`- und das `FIN`-Flag gesetzt sind, dazu verwendet werden, einen Paketfilter zu unterlaufen, der sonst `SYN`-Segmente filtert. Um den klassischen »Phonebook«-Angriff auf Webserver zu erkennen, könnte ein IDS nach Paketen suchen, die die Zeichenkette »`/cgi-bin/phf?`« enthalten und so weiter.

Netzbasierte IDS, die komplette Pakete betrachten, können von einem Angreifer ausgetrickst werden, indem dieser die Pakete »fragmentiert«. Fragmentierung dient eigentlich dazu, IP-Datagramme zu übertragen, die größer sind als die MTU des Schicht-2-Mediums, also nicht auf einen Sitz geschickt werden können. (Ethernet zum Beispiel begrenzt die Nutzdaten in einem Frame auf 1500 Bytes, aber IP-Datagramme können 65535 Bytes lang sein). Die sendende Station schickt die Daten in mehreren Fragmenten, die die empfangende Station (hinter dem IDS) dann wieder zusammensetzt. Daraus folgt, dass das IDS ebenfalls Fragmente zusammensetzen muss, bevor es paketbasierte Regeln anwendet.

In Analogie zur IP-Fragmentierung ist ein TCP-Datenstrom in Segmente aufgeteilt (ein Segment pro IP-Datagramm). Auch hier könnte ein Angreifer einen verdächtigen Datensatz in Form von sehr vielen kleinen Segmenten verschicken und so hoffen, ein IDS zu unterlaufen. Deswegen sollte ein IDS auch TCP-Datenströme aus Segmenten zusammensetzen, bevor es sie analysiert.

Viele Angriffe lassen sich auf unterschiedliche Arten »hinschreiben«, ohne an ihrer Funktion etwas zu ändern. Auch das gibt Angreifern die Möglichkeit, zu versuchen, ein IDS zu verwirren und die Erkennung von Regeln oder Signaturen auszuhebeln. Zum Beispiel läßt der HTTP-Aufruf

```
GET /cgi-bin/phf?alias=x%0a/bin/cat%20/etc/passwd
```

sich auch als

```
GET /%63gi-bin/%70hf?%60lias=x%0a/bin/cat%20/etc/passwd
```

formulieren. Auch solche »Tricks« sollte ein IDS erkennen und rückgängig machen können.

8.4.2 Snort installieren und testen

Die einfachste Methode, Snort zu installieren, besteht wie üblich darin, die entsprechenden Pakete Ihrer Linux-Distribution zu benutzen. Sollte diese Snort nicht enthalten oder die enthaltene Version Ihnen nicht aktuell genug sein, können Sie auf den originalen Quellcode zurückgreifen, den Sie auf `http://www.snort.org/` finden.

Damit Sie mit dem Snort-Quellcode wirklich etwas anfangen können, brauchen Sie außerdem noch einige andere Softwarepakete, etwa die libpcap (von `http://www.tcpdump.org`). Details stehen in der Snort-Dokumentation. Prüfen Sie jeweils, ob Sie zumindest für diese Softwarepakete auf die entsprechenden (Entwicklungs-)Pakete Ihrer Distribution zurückgreifen können, um sich Arbeit zu ersparen.


Wenn Sie den Snort-Quellcode und alle seine Abhängigkeiten zur Verfügung haben, sollte das übliche

```
$ ./configure && make
# make install
```

für den Rest ausreichen.

Außer der Snort-Software brauchen Sie noch Regeln für die Erkennung von Angriffen. Diese werden separat verteilt, damit sie öfter aktualisiert werden können. Einen vernünftigen Grundstock finden Sie ebenfalls auf `www.snort.org`.


Für eine produktive Installation von Snort empfiehlt sich die Installation eines Programms wie `oinkmaster`, das sich automatisch um die Aktualisierung der Regelsätze kümmern kann. Die Details sprengen den Rahmen dieser Einführung.

Einfache Tests Das wichtigste Programm im Snort-Paket heißt (vorhersehbar) `snort`. Im einfachsten Fall verhält es sich ähnlich wie `tcpdump`:

```
# snort -v
```

gibt abgekürzte Informationen über den Datenverkehr auf allen (!) Netzwerkschnittstellen auf dem Bildschirm aus. (Für die Anzeige sorgt eigentlich die Option `-v`).

Mit einem Kommando wie

```
# snort -vde -i eth0
```

betrachtet `snort` gezielt den Datenverkehr auf der Netzwerkschnittstelle `eth0` und gibt die wichtigsten Informationen im Klartext aus. Dabei zeigt die Option `-d` die Paketinhalte und `-e` Ethernet-Informationen an.


Wie bei `tcpdump` können Sie bei `snort` über BPF-Regeln eine Vorauswahl der zu betrachtenden Pakete treffen.

Daten protokollieren Ebenfalls ähnlich wie `tcpdump` kann `snort` den Datenverkehr in einer Datei aufzeichnen. Dazu müssen Sie mit der Option `-l` ein Verzeichnis angeben, in dem die Daten abgelegt werden sollen:

```
# snort -vde -i eth0 -l /var/log/snort
```

In diesem Verzeichnis legt `snort` Unterverzeichnisse an, deren Namen den IP-Adressen der beteiligten Stationen entsprechen. Normalerweise wählt `snort` dafür die Adresse des Clients (es orientiert sich an der hohen Portnummer).


Sie können die Adresse(n) des »Heimnetzwerks« von Snort angeben, damit es als Dateinamen die IP-Adressen von Stationen *außerhalb* dieses Netzes verwendet. Sind beide beteiligten Stationen im Heimnetzwerk, zählt wieder die Adresse der Station mit der höheren Portnummer.

Alternativ können Sie die Daten auch im (binären) pcap-Format protokollieren. Dann bekommen Sie nur eine Datei:

```
# snort -b -l /var/log/snort
```

Die Optionen `-d` und `-e` sind dann nicht mehr nötig – `snort` protokolliert von sich aus die kompletten Pakete.

Der Name der binären Protokolldatei ergibt sich aus einem Zeitstempel, der dem Anfang des Protokolls entspricht, gefolgt von `snort.log`.

Die von `snort` angelegten binären Protokolldateien können Sie anschließend mit jedem Programm weiterverarbeiten, das das pcap-Format versteht. Außer `snort` selbst sind das zum Beispiel `tcpdump` oder Wireshark. Für `snort` etwa müssen Sie Folgendes angeben, um den Inhalt einer Datei namens `snort.log` auszugeben:

```
$ snort -dv -r snort.log
```

Auch hier können Sie über BPF-Regeln bestimmte Pakete vorauswählen. Etwas wie

```
$ snort -dv -r snort.log icmp
```

würde zum Beispiel nur ICMP-Pakete betrachten.

Der Vorteil des binären Protokolls liegt darin, dass Snort es wesentlich schneller schreiben kann als das Standardformat (es muss keine Zeit mit dem Konvertieren der binären Paketformate in lesbaren Text verbringen). Der Verzicht auf die Konsolenausgabe, die wir sonst mit `-v` eingeschaltet haben, ist ein wichtiger weiterer Geschwindigkeitsfaktor.

Sie sollten Snort, wenn es tatsächlich als IDS läuft, niemals mit der Option `-v` betreiben – es ist praktisch garantiert, dass dann Pakete überlesen werden, weil Snort nicht mit dem Datenverkehr Schritt halten kann.

8.4.3 Snort als IDS

Damit Snort tatsächlich Regeln auf die gelesenen Daten anwendet, müssen Sie ihm eine Konfigurationsdatei übergeben, die die Regelsätze beschreibt. Erst dann arbeitet Snort tatsächlich als IDS:

```
# snort -de -l /var/log/snort -c /etc/snort/snort.conf
```

Im IDS-Modus protokolliert Snort nur noch diejenigen Pakete, die tatsächlich von Regeln herausgefiltert wurden. Die Protokollierung erfolgt hier im Klartext (keine Option `-b`).

Die Regeln für die Klartext-Protokollierung entsprechen den eben diskutierten, was die Dateinamenvergabe angeht.

Wenn Snort etwas Verdächtiges im Datenverkehr findet, gibt es einen »Alarm« (engl. *alert*) aus. Der Alarm landet normalerweise in einer relativ ausführlichen Form auf der Konsole.

Mit der Option `-A` können Sie festlegen, wie genau ein Alarm ausgegeben wird. Die Möglichkeiten sind unter anderem:

`full` Die Standardvorgabe – ziemlich langsam und daher für Produktionszwecke nicht wirklich sinnvoll

`fast` Eine an `syslog` angelehnte einzeilige Meldung mit einem Zeitstempel, einer Nachricht und den Quell- und Ziel-Adressen und -Ports

`unsock` Die Nachrichten werden an ein Unix-Domain-Socket geschickt, wo ein anderes Programm sie lesen kann

`console` Wie `fast`, aber auf der Konsole

`none` Überhaupt keine Alarmmeldungen

Mit der Option `-s` können Sie Alarmmeldungen an den Systemprotokolldienst (Syslog) schicken. Normalerweise verwendet Snort die Kategorie `authpriv` und die Priorität `alert`; wenn Sie diese ändern wollen, müssen Sie die Konfigurationsdatei benutzen.

Konfigurationsdatei

Die Konfigurationsdatei von Snort heißt typischerweise `/etc/snort/snort.conf`. Für ihre Syntax gelten die üblichen Grundregeln (Kommentarzeilen mit »#«, Leerzeilen werden ignoriert). Jede Zeile enthält eine Direktive plus eventuelle Parameter.

Mit `include` gefolgt von einem Dateinamen können Sie eine benannte Datei an der betreffenden Stelle in die Konfiguration aufnehmen.

Variable

Eine wichtige Klasse von Direktiven beschäftigt sich mit Variablen. Diese dienen zur zentralen Definition von Parametern wie IP-Adressen, Ports oder Pfadangaben:

```
var RULES_PATH rules/
ipvar HOME_NET 192.168.0.0/24
ipvar EXTERNAL_NET any
ipvar DNS_SERVERS 192.168.0.1
ipvar HTTP_SERVERS $DNS_SERVERS
portvar HTTP_PORTS [80,8080]
```

Auf den Wert einer Variablen greifen Sie wie in der Shell durch ein vorgesetztes »$« zu. Um Missverständnisse zu vermeiden, können Sie den Variablennamen auch in »`$(…)`« setzen.

Außerdem gibt es ein paar Tricks, die Ihnen vage von der Shell bekannt vorkommen könnten: »`$(var:-default)`« liefert den Inhalt der Variablen `var` oder die Zeichenkette `default`, falls `var` undefiniert ist. »`$(var:?meldung)`« liefert entweder den Inhalt der Variablen `var` oder gibt, falls `var` undefiniert ist, die Meldung `meldung` aus und bricht das Programm ab:

```
ipvar HOME_NET 192.168.1.0/24
log tcp any any -> $(HOME_NET:?HOME_NET ist undefiniert!) 23
```

Die Direktive `ipvar` dient zur Definition von IP-Adressen und `portvar` zur Definition von Ports. Alle anderen Variablen werden mit `var` definiert.

IP-Adressen können einzeln, als Listen, als CIDR-Blöcke (mit Netzmaske) oder als Kombination dieser Möglichkeiten angegeben werden. Außerdem steht das Schlüsselwort `any` für »jede beliebige Adresse«. Listen stehen in eckigen Klammern:

```
[10.0.0.1,192.168.4.0/24,![192.168.4.2,192.68.4.3]]
```

Diese Liste passt auf die Adresse `10.0.0.1` und die Adressen, die mit `192.168.4` anfangen, bis auf `192.168.4.2` und `192.168.4.3`. Das »`!`« steht für Negation. – Die Regel für Negation in einer Liste ist, dass zunächst alle nicht negierten Elemente der Liste ODER-verknüpft werden. Anschließend werden aus diesem Zwischenergebnis alle negierten Elemente wieder entfernt.

Ports können ebenfalls einzeln, als Listen und als Bereiche angegeben werden. Listen stehen in eckigen Klammern und Bereiche verwenden »`:`« als Trennzeichen, etwa in

```
[10:50,888:900,!888:889]
```

(Auch hier steht das »`!`« für Negation, und `any` steht für »beliebige Ports«.)

Früher konnten Sie alle Arten von Variablen mit `var` definieren. Das ist inzwischen verpönt – `var` statt `ipvar` wird zwar noch unterstützt, genau wie `var` statt `portvar`, falls der Name der Variable mit `PORT_` anfängt oder mit `_PORT` aufhört, aber das wird nicht immer so bleiben.

Ein paar Sachen sind komplett verboten, namentlich »`!any`« und andere logische Widersprüchlichkeiten wie zum Beispiel

```
portvar WEIRD_IP [1.1.1.0/24,!1.1.0.0/16]
portvar WEIRD_PORTS [80,!80]
```

Konfigurationsparameter

Die `config`-Direktive dient zum Einstellen von wichtige Snort-Konfigurationsparametern. Hier ein paar Beispiele:

```
config interface: eth0
config set_gid: snort
config set_uid: snort
config alertfile: /var/log/snort/alerts
config daemon                                    Als Hintergrunddienst laufen
```

Die Liste der erlaubten Parameter ist lang und der Snort-Dokumentation zu entnehmen.

Snort kann zur Laufzeit mit zusätzlichen Modulen erweitert werden. Die Direktive

```
dynamicpreprocessor directory /usr/lib/snort_dp
```

liest alle Dateien im Verzeichnis `/usr/lib/snort_dp` als »dynamische Präprozessoren« (dazu gleich mehr). Mit »`dynamicpreprocessor file` …« können Sie gezielt eine einzelne Datei laden. Analog gibt es Direktiven wie `dynamicengine` oder `dynamicdetection`.

Präprozessoren

Zur Vorverarbeitung von Daten verwendet Snort konfigurierbare Präprozessoren. Diese kümmern sich um Aufgaben wie das Defragmentieren von Rohdaten oder die Identifikation spezieller Elemente eines Kommunikationsvorgangs. Zu den gängigen Präprozessoren gehören zum Beispiel:

`frag3` zur zielbezogenen IP-Defragmentierung. »Zielbezogen« heißt, dass bei der Defragmentierung die Eigenheiten verschiedener Betriebssysteme berücksichtigt werden können. Wenn ein Angreifer also Daten auf eine bestimmte Weise fragmentiert, um Probleme im konkreten Betriebssystem eines Zielrechners auszunutzen, dann würde ein IDS, das nicht genau dasselbe Verfahren zur Defragmentierung verwendet, einen entsprechenden Angriff möglicherweise übersehen. Snort mit `frag3` versucht – mit Informationen über die lokalen Stationen – diese Hürde zu umgehen.

`stream5` zur Rekonstruktion von TCP- und UDP-Datenströmen und Verfolgung von Verbindungen. TCP-Sitzungen werden trivial identifiziert und UDP-»Sitzungen« auf der Basis der beteiligten Adressen und Portnummern. Wie `frag3` nimmt `stream5` Rücksicht auf die speziellen TCP/IP-Implementierungen der Zielsysteme.

`sfPortscan` erkennt Portscans, die als Vorstufe zu einem tatsächlichen Angriff ausgeführt werden. Es nimmt besondere Rücksicht auf `nmap` (Abschnitt 6.2), aber versucht auch, »verteilte« Portscans zu identifizieren, bei denen verschiedene Rechner als Gegenstellen fungieren, sowie »Portsweeps«, bei denen ein Rechner versucht, denselben Port auf einer Reihe von Stationen zu testen.

`perfmonitor` liefert ausführliche Statistiken über den Betrieb von Snort.

http_inspect dekodiert und normalisiert HTTP-Daten. Damit ist es möglich, Regeln zu formulieren, die auf Eigenschaften einer HTTP-Anfrage oder -Antwort Bezug nehmen. Der Präprozessor kann optional Rücksicht auf die Besonderheiten populärer Webserver wie Apache oder IIS nehmen.

smtp dekodiert ähnlich wie `http_inspect` SMTP-Daten für die Weiterverarbeitung in Regeln. Dabei werden auch Protokolleigenschaften geprüft, etwa die Einhaltung maximaler Zeilenlängen.

arpspoof erkennt Angriffe auf das ARP und Inkonsistenzen bei der Abbildung von Ethernet-MAC-Adressen auf IP-Adressen.

(Eine ausführliche Liste mit allen Konfigurationseinstellungen steht im Snort-Handbuch.)

Präprozessoren werden über die `preprocessor`-Direktive konfiguriert. Für den SMTP-Präprozessor könnte das zum Beispiel so aussehen:

```
preprocessor smtp: \
  ports { 25 587 } \
  inspection_type stateful \
  normalize_cmds { EXPN VRFY RCPT } \
  alt_max_command_line_len 260 { MAIL }
```

Das heißt, der Präprozessor soll Verbindungen zu den Ports 25 (`smtp`) und 587 (`submission`) betrachten. Dabei wird der Verbindungsstatus über mehrere Datenpakete hinweg verfolgt (`inspection_type stateful`). Die Kommandos `EXPN`, `VRFY` und `RCPT` werden »normalisiert«, das heißt, es wird geprüft, ob hinter dem Kommando mehr als ein Leer- oder Tab-zeichen steht, und `MAIL`-Kommandozeilen dürfen höchstens 260 Zeichen lang sein.

Regelsätze

Der eigentlich interessante Teil der Snort-Konfiguration sind die Regelsätze. Wie wir schon erwähnt haben, müssen Sie diese getrennt von der eigentlichen Software beziehen. Welchen Regelsatz (oder welche Regelsätze) Sie einsetzen sollten, hängt von Ihren Ansprüchen (und Ihrem Geldbeutel) ab:

- Der aktuellste Regelsatz ist der offizielle »VRT-zertifizierte« Regelsatz von Sourcefire, der Firma, die sich um die Weiterentwicklung von Snort kümmert. (VRT steht für »Vulnerability Research Team«). Um diesen tagesaktuell einsetzen zu können, müssen Sie ein kostenpflichtiges Abonnement erwerben. Für persönliche oder Ausbildungszwecke kostet das etwa 30 US-Dollar im Jahr; für kommerziellen Einsatz müssen Sie (oder Ihre Firma) deutlich tiefer in die Tasche greifen.
- Registrierte Benutzer von `snort.org` dürfen dieselben Regeln kostenlos benutzen, allerdings tauchen Änderungen und Erweiterungen dort erst mit einer Verzögerung von 30 Tagen auf.
- Schließlich gibt es auf `www.snort.org` noch ein unter der GPL lizenziertes »Community Ruleset«. Dieses wird täglich aktualisiert und ist eine Teilmenge des VRT-zertifizierten Regelsatzes; Anwender des kostenlosen um 30 Tage verzögerten Regelsatzes können das Community Ruleset zusätzlich installieren, um Zugriff auf aktuellere Regeln zu haben.

Regel-Syntax

Snort-Regeln selbst bestehen immer aus einem »Kopf« und zusätzlichen Optionen. Der Kopf beschreibt die beteiligten Stationen und Portnummern und gibt eine Aktion an, die für »passende« Datenpakete unternommen werden soll. Die Optionen beschreiben die gesuchten Datenpakete genauer. Hier ist ein Beispiel aus dem Snort-Handbuch:

```
alert tcp any any -> 192.168.1.0/24 111 \
  (content:"|00 01 86 a5|"; msg:"mountd access";)
```

Diese Regel löst einen Alarm aus, wenn irgendein Rechner über eine TCP-Verbindung zum Port 111 irgendeines Rechners im Netz `192.168.1.0/24` die Bytefolge »`00 01 86 a5`« schickt. Dies wird als versuchter Zugriff auf den NFS-`mountd` gewertet und entsprechend gemeldet. Der Text bis zur linken runden Klammer ist der Kopf; der Teil innerhalb der Klammern beschreibt die Optionen. Optionen bestehen immer aus einem Wort gefolgt von einem Doppelpunkt, allfälligen Parametern und einem Semikolon.

Alle »Bedingungen« in einer Regel (Adressen und Ports im Kopf und Optionen, die weitere Bedingungen beschreiben) werden implizit logisch UND-verknüpft. Alle definierten Regeln werden implizit logisch ODER-verknüpft.

Aktionen

Die folgenden Aktionen stehen zur Verfügung:

`alert` Einen Alarm auslösen (gemäß der definierten Methode) und das Paket protokollieren

`log` Das Paket nur protokollieren (ohne Alarm)

`pass` Das Paket ignorieren

`activate` Wie `alert`; zusätzlich wird eine weitere »dynamische« Regel aktiviert

`dynamic` Nichts tun, bis die Regel von einer `activate`-Regel »scharf geschaltet« wird, dann wie `log`

`drop` Das Paket protokollieren und verwerfen

`reject` Das Paket protokollieren, verwerfen und dann dem Absender ein TCP-`RST` (für TCP-Verbindungen) oder eine ICMP-»Port unerreichbar«-Nachricht (für UDP) schicken

`sdrop` Das Paket verwerfen, aber nicht protokollieren

Die letzten drei Aktionen ergeben natürlich nur dann Sinn, wenn der Rechner, auf dem Snort läuft, tatsächlich Bestandteil der Verbindung zwischen Absender und Empfänger der Pakete ist, also nicht, wenn das IDS nur über einen »passiven Tap« angebunden ist.

Sie können sich auch Ihre eigenen Aktionen definieren: Mit etwas wie

```
ruletype redalert
{
   type alert
   output alert_syslog: LOG_AUTH LOG_ALERT
   output log_tcpdump: alert.log
}
```

können Sie zum Beispiel Regeln der Form

```
redalert tcp any any -> ◁◁◁◁◁
```

verwenden, die – falls sie auslösen – eine Nachricht ins Systemprotokoll schreiben und das betreffende Paket im Binärformat in der Datei `alert.log` abspeichern.

Protokoll

Der Rest des Regelkopfs beginnt mit dem zu betrachtenden Protokoll – Snort versteht `tcp`, `udp`, `icmp` und `ip`. Danach kommen die Adressen und ggf. Portnummern.

Die Syntax für Adressen und Portnummern folgt den oben für die Direktiven `ipvar` und `portvar` beschriebenen Regeln.

Eine Regel wie

```
log tcp any 1024: -> 192.168.1.0/24 80
```

würde also auf TCP-Zugriffe irgendwelcher Rechner mit Portnummern ab 1024 auf den Port 80 lokaler (?) Rechner passen.

Sie können der Übersicht halber hier natürlich Variable benutzen:

```
ipvar HOME_NET 192.168.1.0/24
portvar CLIENT_PORTS 1024:
portvar HTTP_PORT 80

log tcp any $CLIENT_PORTS -> $HOME_NET $HTTP_PORT
```

Auf den ersten Blick macht das alles zwar nur umständlicher, aber wenn Sie die Adressen und Ports in mehreren Regeln verwenden müssen, werden Ihre Regelsätze dadurch weitaus wartungsfreundlicher.

Richtungsoperator

Das »->« zwischen der Quell- und der Zielangabe beschreibt die Flussrichtung der Daten. Insbesondere steht es dafür, dass nur eine Richtung der Kommunikation analysiert oder protokolliert wird. Um beide Richtungen zu erfassen, müssen Sie den Operator »<>« verwenden:

```
log tcp !192.168.1.0/24 any <> 192.168.1.0/24 110
```

Beachten Sie, dass es keinen »<-«-Operator gibt. Der Sinn dahinter ist, dass das zu unleserlichen Regelsätzen führen würde.

Dynamisch aktivierte Regeln

Dynamisch aktivierte Regeln treten nur dann in Kraft, wenn sie von anderen Regeln freigeschaltet wurden. Ihr Hauptzweck besteht in der »Beweissicherung«, wenn eine andere Regel ausgelöst wurde. Die andere Regel verwendet eine `activates`-Option, um zu bestimmen, welche dynamische Regel aktiviert werden soll; die dynamische Regel muss eine dazu passende `activated_by`-Option haben:

```
activate tcp !$HOME_NET any -> $HOME_NET 143 \
    (flags:PA; \                                    PSH und ACK gesetzt?
     content:"|E8C0FFFFFF|/bin"; activates:1; \
     msg:"IMAP buffer overflow!";)
dynamic tcp !$HOME_NET any -> $HOME_NET 143 \
    (activated_by:1; count:50;)
```

Wenn die erste Regel im Beispiel einen IMAP-Pufferüberlauf feststellt, aktiviert sie die zweite Regel. Diese sorgt dafür, dass die ersten 50 Pakete der Konversation protokolliert werden. Diese können Sie dann später analysieren – höchstwahrscheinlich steht etwas Interessantes drin.

Optionen

Snort unterstützt eine große Menge an Optionen, deren Details Sie am besten der Snort-Dokumentation entnehmen. Hier nur ein kurzer Überblick über die wichtigsten Möglichkeiten.

IP-Datagramme

Für IP-Datagramme stehen Ihnen zum Beispiel die folgenden Optionen zur Verfügung:

`tos`, `ttl`, `ip_proto` Testet die *type of service*-, *time to live*- und Protokollfelder des IP-Datagrammkopfs

`sameip` Prüft, ob Absender- und Empfängeradresse identisch sind

`ipopts` Testet IP-Optionen, etwa `ssrr` für Source-Routing (die sich aber nicht in weitem Gebrauch befinden); Details stehen in der Snort-Dokumentation

fragbits Testet die Fragmentierungs-Bits im Kopf des IP-Datagramms; Details stehen in der Snort-Dokumentation

dsize Prüft die Größe des Datagramms, etwa um Datagramme zu identifizieren, die Pufferüberläufe auslösen könnten:

```
dsize:1024;                    Genau 1024 Oktette
dsize:>1024;                   Mehr als 1024 Oktette
dsize:<1024;                   Weniger als 1024 Oktette
dsize:1024<>32768;             Zwischen 1024 und 32768 Oktette
```

Da die Datenpakete von ICMP, TCP und UDP alle auch IP-Datagramme sind, können Sie diese Optionen natürlich auch dort benutzen.

ICMP Hier sind einige Optionen für den Gebrauch mit ICMP:

itype, icode Testet den ICMP-Typ und -Code des betreffenden Datagramms

icmp_id, icmp_seq Testet ICMP-ECHO-Identifikation und -Folgenummer (nicht völlig abwegig; manche Programme nutzen diese Felder als verdeckten Kommunikationskanal)

TCP Und hier ein paar für TCP:

flags Testet die TCP-Flags: Mögliche Werte sind `F` (`FIN`), `S` (`SYN`), `R` (`RST`), `P` (`PSH`), `A` (`ACK`), `U` (`URG`), `C` (`CWR`) und `E` (`ECE`). Sie können diese beliebig kombinieren und außerdem noch Kriterien angeben: Ein vorgestelltes `*` steht für eine ODER-Verknüpfung (Standard ist UND), ein `+` läßt weitere gesetzte Flags außer den Genannten zu, und ein `!` sorgt dafür, dass die Option passt, wenn die genannten Flags *nicht* gesetzt sind. Die Option `flags` hat einen zweiten Parameter, in dem Sie optional angeben können, welche Flags ignoriert werden sollen. Die Regel

```
alert tcp any any -> any any (flags:SF,CE;)
```

(aus dem Snort-Handbuch) passt zum Beispiel auf TCP-Segmente, die sowohl das `SYN`- als auch das `FIN`-Flag gesetzt haben, wobei die Werte des `CWR`- und des `ECE`-Flags egal sind.

seq Prüft auf eine bestimmte TCP-Folgenummer (nicht furchtbar nützlich).

ack Prüft auf eine bestimmte TCP-Bestätigungsnummer (auch nicht furchtbar nützlich).

Inhalt Den Inhalt von Paketen untersuchen Sie mit Optionen wie den folgenden:

content definiert eine zu suchende Bytefolge. Diese wird entweder als ASCII-Text angegeben (Groß- und Kleinschreibung wird beachtet!) oder als hexadezimale Binärdaten in »|…|«. Die Darstellung darf auch gemischt werden:

```
content:"|00|xyz|01|uvw"       Byte 00, Text »xyz« usw.
```

In einer Regel dürfen mehrere `content`-Optionen auftauchen. Wie alle Optionen werden diese miteinander UND-verknüpft.

Folgt der `content`-Option eine `nocase`-Option, wird bei dieser `content`-Option Groß- und Kleinschreibung *nicht* beachtet.

Folgt der `content`-Option eine `depth`-Option mit einem ganzzahligen Parameter, dann schaut Snort bei der Suche nur die ersten Bytes des Inhalts an:

```
content:"Q"; depth:10;         Suche »Q« in den ersten 10~Bytes
```


Folgt der `content`-Option eine `offset`-Option mit einem ganzzahligen Parameter, dann fängt Snort erst dort an zu suchen:

```
content:"cgi-bin/phf"; offset:4; depth:20;
```


Es gibt noch ein paar weitere Optionen dieser Art, die den Suchbereich einschränken. Schauen Sie in die Snort-Dokumentation.


Mit nachgestellten Optionen wie `http_client_body`, `http_cookie` oder `http_header` können Sie die Suche auf bestimmte Teile einer HTTP-Anfrage anwenden. Dafür müssen Sie natürlich den entsprechenden Präprozessor aktiviert haben. Auch hierzu hat die Snort-Dokumentation interessante Informationen zu bieten.

`uricontent` durchsucht den normalisierten Inhalt des URI-Felds einer HTTP-Anfrage (und entspricht damit `content` mit einer `http_uri`-Option). »Normalisiert« bedeutet, dass Sequenzen wie »`%2f`« durch ihr ASCII-Äquivalent ersetzt wurden (das macht der Präprozessor). Verwenden Sie sie also nicht in Regeln, denn die passen dann nie.


Dasselbe gilt für »Verzeichnistraversierung« mit »`/../`«. Ein URI wie

```
/cgi-bin/abcxyz123abcxyz123/..%252fp%68f?
```

wird »normalisiert« also zu

```
/cgi-bin/phf?                          Telefonbuch-Angriff; etwas verstaubt
```


An den nicht normalisierten Inhalt kommen Sie bei Bedarf mit `content` ran, indem Sie die nachgestellte Option `http_raw_uri` benutzen.

`pcre` verwendet Perl-kompatible reguläre Ausdrücke zur Suche (siehe `http://www.pcre.org`). Dies ist eine relativ »teure« Operation; es lohnt sich also meistens, außer einer `pcre`-Option auch noch eine `content`-Option zu haben, die eine Voruntersuchung vornimmt. Pakete, auf die schon die `content`-Option nicht passt, müssen nicht mit der aufwendigen PCRE-Suche beackert werden:

```
alert tcp any any -> any 80 ( \
  content:"/foo.php?id="; \                                  Vorkontrolle
  pcre:"/foo.php?id=[0-9]1,10/iU";)                          Genauere Prüfung
```

`flow` ist eigentlich keine inhaltsbezogene Option. Statt dessen erlaubt sie es, zusätzliche Bedingungen für TCP-Verbindungen aufzustellen (Vorbedingung ist die Verwendung des `stream5`-Präprozessors). Damit können Sie zum Beispiel zwischen HTTP-Clients in Ihrem Netz und HTTP-Servern in Ihrem Netz unterscheiden. Hier die wichtigsten Bedingungen:

`from_client`, `to_server` Passt nur auf Client-Anfragen

`to_client`, `from_server` Passt nur auf Server-Antworten

established Passt nur auf etablierte TCP-Sitzungen (nach dem Drei-Wege-Handshake)

not_established Passt nur auf anfängliche Pakete einer TCP-Sitzung

(Es gibt noch ein paar weitere Möglichkeiten.) Sie können mehrere der Bedingungen mit Kommas getrennt aneinanderreihen und so UND-verknüpfen.

`detection_filter` ist auch keine inhaltsbezogene Option, sondern kann prüfen, ob eine Regel in einer gewissen Zeitspanne »oft« ausgelöst wurde. Damit können Sie zum Beispiel Angriffe auf die Secure Shell erkennen: Die Regel

```
drop tcp any any -> 192.168.1.0/24 22 ( \
  msg:"SSH Brute Force Attempt"; \
  flow:established,to_server; \
  content:"SSH"; nocase; offset:0; depth:4; \
  detection_filter:track by_src, count 10, seconds 30; )
```

spricht an, wenn innerhalb von 30 Sekunden 10 Login-Versuche unternommen wurden.

Zu guter Letzt gibt es noch Optionen, die sich im weitesten Sinne um die Ausgabe von Nachrichten kümmern:

msg gibt die Nachricht an, die ausgegeben werden soll, wenn die Regel anschlägt.

logto schreibt alle Pakete, die die Regel auslösen, in die angegebene Datei (allerdings nur bei textbasierter Protokollierung, nicht bei binärer).

reference erlaubt Verweise auf externe Kataloge von Angriffen oder Sicherheitslücken. Snort hat eine hartkodierte Liste von Katalogen, die jeweils mit einem URL-Präfix korrespondieren. `cve` ist zum Beispiel eine Abkürzung für »`http://cve.mitre.org/cgi-bin/cvename.cgi?name=`«. Hier ist ein Beispiel aus der Snort-Dokumentation:

```
alert tcp any any -> any 21 ( \
  msg:"IDS287/ftp-wuftp260-venglin-linux"; \
  flags:AP; content:"|31c031db 31c9b046 cd80 31c031db|"; \
  reference:arachnids,IDS287; reference:bugtraq,1387; \
  reference:cve,CAN-2007-1574;)
```

sid identifiziert die Snort-Regel eindeutig über eine Nummer. Nummern von 100 bis 999.999 sind reserviert für offizielle Regeln; Ihren eigenen Regeln sollten Sie Nummern ab 1.000.000 geben (nicht besonders tippfreundlich, aber was will man machen).

rev dient dazu, einer Regel eine Versionsnummer zu geben. Sie sollten Ihre Regeln immer mit einer Nummer und einer Versionsnummer ausstatten.

classtype ordnet eine Regel einer Angriffsklasse zu. Damit lassen sich die Ausgabedaten von Snort besser organisieren. Eine Regel wie

```
alert tcp any any -> any 25 (msg:"SMTP EXPN root"; flags:A+; \
    content:"expn root"; nocase; classtype:attempted-recon;)
```

(aus der Snort-Dokumentation) betrachtet einen Treffer als `attempted-recon`, also »versuchte Ausspähung«. Snort liefert eine Reihe von Angriffsklassen mit (siehe Tabelle 8.1); Sie können sich mit »`config classification`« Ihre eigenen definieren.

priority gibt einer Regel eine (numerische) Priorität. Diese überschreibt die Standardpriorität einer der Regel möglicherweise zugeordneten Angriffsklasse.

resp erlaubt es, eine weitergehende Reaktion zu definieren, falls die Regel anspricht. Dabei handelt es sich typischerweise um einen mehr oder weniger gewaltsamen Verbindungsabbruch.

react erlaubt es, eine HTML-Seite über eine Verbindung zu schicken und sie dann abzubrechen. Die HTML-Seite kann aus einer benannten Datei kommen. Konsultieren Sie die Snort-Dokumentation für die Details.

Beispiele

Hier zum Abschluss noch ein paar Beispiele aus den offiziellen Regelsätzen:

Tabelle 8.1: Snort-Angriffsklassen

classtype	Beschreibung	Priorität
attempted-admin	Versuch, Administratorprivilegien zu erlangen	hoch
attempted-user	Versuch, Benutzerprivilegien zu erlangen	hoch
inappropriate-content	Anstößiger Inhalt wurde entdeckt	hoch
policy-violation	Potentielle Verletzung einer Firmenregel	hoch
shellcode-detect	Ausführbarer Code gefunden	hoch
successful-admin	Administratorprivilegien wurden erlangt	hoch
successful-user	Benutzerrechte wurden erlangt	hoch
trojan-activity	Ein Netzwerk-Trojaner wurde entdeckt	hoch
unsuccessful-user	Vergeblicher Versuch, Benutzerrechte zu erlangen	hoch
web-application-attack	Angriff auf eine Webanwendung	hoch
attempted-dos	Versuch eines *denial of service*	mittel
attempted-recon	Versuchtes Ausspähen von Informationen	mittel
bad-unknown	Möglicherweise schädlicher Datenverkehr	mittel
default-login-attempt	Angriffsversuch mit Standard-Benutzerdaten	mittel
denial-of-service	*Denial-of-service*-Angriff entdeckt	mittel
misc-attack	Unspezifischer Angriff	mittel
non-standard-protocol	Unübliches Protokoll oder Protokollverletzung	mittel
rpc-portmap-decode	RPC-Anfrage gefunden	mittel
successful-dos	Erfolgreicher *Denial-of-service*-Angriff	mittel
successful-recon-largescale	Informationsleck in großem Umfang	mittel
successful-recon-limited	Begrenztes Informationsleck	mittel
suspicious-filename-detect	Verdächtiger Dateiname gefunden	mittel
suspicious-login	Anmeldeversuch mit einem verdächtigen Benutzernamen gefunden	mittel
system-call-detect	Systemaufruf wurde gefunden	mittel
unusual-client-port-connection	Client benutzte einen unüblichen Port	mittel
web-application-activity	Zugriff auf eine möglicherweise verwundbare Web-Anwendung	mittel
icmp-event	ICMP-Vorgang	niedrig
misc-activity	Unspezifische Aktivität	niedrig
network-scan	Netzwerk-Scan entdeckt	niedrig
not-suspicious	Unverdächtiger Datenverkehr entdeckt	niedrig
protocol-command-decode	Generische Protokollkommando-Dekodierung	niedrig
string-detect	Verdächtige Zeichenkette gefunden	niedrig
unknown	Unbekannter Datenverkehr	niedrig
tcp-connection	TCP-Verbindung entdeckt	sehr niedrig

(Aus dem Snort-Handbuch.) Dass eine Angriffsklasse in dieser Tabelle existiert, heißt *nicht*, dass Snort in der Lage ist, alle möglichen (oder überhaupt irgendwelche) solchen Angriffe zu erkennen. Die Angriffsklassen dienen lediglich zur Strukturierung der Regelsätze.

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 \
  (msg:"BAD-TRAFFIC tcp port 0 traffic"; flow:stateless; \
   classtype:misc-activity; sid:524; rev:8;)
```

Der TCP-Port 0 ist im wirklichen Leben nicht benutzbar. Ebensowenig wollen wir auf dem Netz die Adresse `127.0.0.1` (oder ihre weitläufige Verwandtschaft) sehen:

```
alert ip any any <> 127.0.0.0/8 any \
  (msg:"BAD-TRAFFIC loopback traffic; \
   reference:url,rr.sans.org/firewall/egress.php; \
   classtype:bad-unknown; sid:528; rev:5;)
```

Die letzte Regel illustriert einen Angriff, bei dem der SMTP-Server überredet werden soll, die Mitglieder einer Verteilerliste aufzuzählen (etwa um die Adressen später mit Spam zumüllen zu können):

```
alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25 \
  (msg:"SMTP expn *@"; flow:to_server,established; \
   content:"expn"; nocase; content:"*@"; \
   pcre:"/^expn\s+*@/smi"; reference:cve,1999-1200; \
   classtype:misc-attack; sid:1450; rev:5;)
```

Beachten Sie hier die Kombination aus `content`- und `pcre`-Optionen.

Snort ist ein extrem umfangreiches und mächtiges Paket, von dem wir hier nur die Grundlagen präsentieren können. Besuchen Sie gegebenenfalls die Webseiten auf `http://www.snort.org/` und lesen Sie vor allem das dort verfügbare, sehr ausführliche Handbuch und die Tutorien.

Übungen

(Alle Übungen setzen ein installiertes Snort voraus. Verwenden Sie den kostenlosen VRT-Regelsatz oder, wenn Sie sich nicht auf `snort.org` registrieren wollen, den frei verfügbaren Community-Regelsatz.)

8.5 [2] Suchen Sie sich 5 Angriffskategorien aus Tabelle 8.1 aus und finden Sie in Ihrem Snort-Regelsatz Beispiele für Regeln, die solche Angriffe erkennen sollen.

8.6 [2] Geben Sie eine Snort-Regel an, die Zugriffe auf eine Webseite namens `http://www.example.com/bla.html` erkennt und (a) die Verbindung kommentarlos abbricht; (b) die Verbindung abbricht und eine Nachricht (als HTML) zurückschickt; (c) die nächsten 20 Pakete derselben Verbindung protokolliert. (Gesucht sind drei verschiedene Regeln.)

8.7 [3] Kampf der Giganten: Installieren Sie (am besten in einer geeigneten Virtualisierungsumgebung) einen Server, der einige einfache Netzwerkdienste anbietet (SMTP, HTTP, …). Installieren Sie Snort auf demselben Rechner oder in einer eigenen virtuellen Maschine (letzteres ist instruktiver, aber etwas aufwendiger). Machen Sie von einem dritten Rechner (am einfachsten dem physikalischen Virtualisierungs-Server) einige `nmap`-Scans und prüfen Sie, ob Snort sie erkennt. Können Sie mit `nmap` einen Portscan durchführen, den Snort nicht entlarvt?

Kommandos in diesem Kapitel

fail2ban-client Sperrt Rechner, von denen Kennwort-Rateangriffe ausgehen (Client) fail2ban-client(8) 122

fail2ban-server Sperrt Rechner, von denen Kennwort-Rateangriffe ausgehen (Server) fail2ban-server(8) 122

snort Netzwerk-Angriffserkennungssystem snort(8) 127

Zusammenfassung

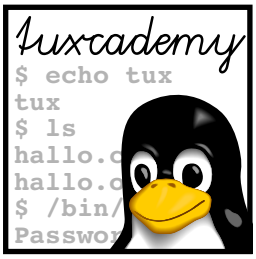
- Netzbasierte Angriffserkennungssysteme versuchen verdächtige Aktivität zu erkennen, indem sie den Datenverkehr im Netz analysieren.
- Die wesentlichen Probleme netzbasierter Angriffserkennungssysteme sind einerseits »falsche Positive« und andererseits die Tatsache, dass sie erst greifen, wenn die normalen Abwehrmechanismen bereits versagt haben.
- `scanlogd` ist ein einfaches Programm zur Erkennung von Portscans.
- `fail2ban` kann einzelne Rechner sperren, wenn von ihnen aus zu oft versucht wurde, auf Dienste wie SSH zuzugreifen.
- Snort ist ein mächtiges netzbasiertes Angriffserkennungssystem.

Literaturverzeichnis

Bei13 Richard Beijtlich. *The Practice of Network Security Monitoring*. No Starch Press, 2013. ISBN 978-1593275099. http://nostarch.com/nsm

CG04 Kerry Cox, Christopher Gerg. *Managing Security with Snort and IDS Tools*. Sebastopol, CA: O'Reilly Media, 2004. ISBN 978-0596006617. http://shop.oreilly.com/product/9780596006617.do

Sch04 Bruce Schneier. *Secrets & Lies – IT-Sicherheit in einer vernetzten Welt*. Weinheim: Wiley/VCH, 2004. ISBN 3-527-50128-2. http://www.schneier.com/book-sandl.html



9

Virtuelle private Netze mit OpenVPN

Inhalt

Lernziele

- Die Grundzüge von VPNs verstehen
- Ein VPN mit OpenVPN einrichten können

Vorkenntnisse

- Allgemeine TCP/IP-Kenntnisse
- Erstellung von X.509-Zertifikaten mit OpenSSL (Anhang B)
- Grundkenntnisse über Kryptografie sind sinnvoll

9.1 Warum VPN?

Die Zeiten, in denen Sie sich im Internet bewegen konnten, ohne an Sicherheit, Verschlüsselung und Authentisierung zu denken, sind vorbei. Zum einen sind die Angriffsziele durch die Kommerzialisierung des Internet lohnender geworten – Kreditkarten-Daten, Online-Banking usw. –, zum anderen haben einschlägige Werkzeuge eine solche Verbreitung gefunden, dass selbst Benutzer ohne technischen Hintergrund Kennwörter auspionieren und schlecht gesicherte Daten entschlüsseln können.

Mit dem Siegeszug moderner kryptografischer Verfahren hat sich die Situation deutlich verbessert, allerdings um den Preis erhöhter Komplexität. Sichere Verbindungen zu implementieren ist ausgesprochen schwierig, da es mit der Verschlüsselung allein nicht getan ist. Neben der Geheimhaltung der Daten kommen die Integrität (Verfälschungssicherheit) der Daten und die Authentisierung des Gegenübers ins Spiel. Das macht die verwendeten Protokolle und Programme aufwendig, was zu weiteren Problemen führt: Komplizierte Programme haben mehr Fehler als einfache Programme, und komplizierte Protokolle haben mehr Fallstricke als einfache Protokolle: Was nützt Ihnen eine sichere Verschlüsselung, wenn Sie nicht sicherstellen können, dass kein Angreifer den notwendigen Schlüssel-Austausch sabotiert?

unsicheres Medium

Um die Verbindung zweier Rechner über ein unsicheres Medium (Client und Server über das Internet oder ein unsicheres Intranet) abzusichern, gibt es grundsätzlich drei Möglichkeiten. Die Sicherung kann auf Anwendungsebene, auf Transportebene oder auf Netzwerkebene erfolgen.

Anwendungsebene

Beim ersten Fall, der Absicherung auf Anwendungsebene, sind die Anwendungen für die Verschlüsselung (und Integrität usw.) verantwortlich. Ein Beispiel hierfür ist SSH als sicherer Ersatz für TELNET, ein anderes TLS, wenn es zum Beispiel innerhalb von SMTP aufgerufen wird. Der relativ leichten Einrichtung von Verschlüsselung auf Anwendungsebene steht die mangelhafte Interoperabilität entgegen: ein altes Programm, das intern das TELNET-Protokoll benutzt, können Sie so nicht nachträglich sicher machen. Außerdem spricht sich die Existenz von SSH nur sehr langsam bei Firmen wie Microsoft und kommerziellen Anbietern von Unix herum.

Transportebene

Um eine Modifikation von Anwendungen zu vermeiden, bietet sich eine auf Transportebene an. Geläufigstes Beispiel hierfür ist HTTPS, bei dem *ad hoc* ein sicherer SSL/TLS-Tunnel aufgebaut wird, durch den dann normales HTTP übertragen wird. Genau so funktioniert auch das nicht mehr häufig eingesetzte SMTPS: erst wird der Tunnel aufgebaut, dann wird SMTP gesprochen. Zwar ist bei HTTPS der Tunnelaufbau üblicherweise fest in Server und Client integriert, es ist aber durchaus möglich, Tunnel und eigentliches Protokoll zu trennen. Das Programm `stunnel` zum Beispiel erlaubt es Ihnen, beliebige TCP-basierte Protokolle abzusichern, indem Sie sie über einen SSL/TLS-Tunnel leiten. Auch die Portweiterleitung der Secure Shell (OpenSSH) leistet das.

Der Vorteil von OpenSSH ist, dass Sie keine besondere Konfiguration auf der Serverseite brauchen, solange dort ein OpenSSH-Server installiert ist.

Entkopplung

Die Entkopplung von Anwendungsprotokoll und verschlüsseltem Tunnel hat viele Vorteile: Das Rad muss nicht mit jeder Anwendung neu erfunden werden, insbesondere profitieren Anwendungen damit von der jeweils besten verfügbaren Tunnel-Lösung. Die Fehlersuche wird deutlich vereinfacht – mit entsprechenden Werkzeugen können Sie den Tunnel von Hand einrichten, danach geht es nur noch um das Original-Protokoll, das häufig textbasiert ist. (Wird hingegen die Verschlüsselung mit der Anwendung verzahnt, wie bei ESMTP mit STARTTLS, so benötigen Sie ein Werkzeug, des die protokolltypische Form versteht, in der hier der Tunnel aufgebaut wird – bei SMTP das Programm `swaks` oder neuere Versionen von `openssl`.)

Nachteile

Als Nachteile müssen Sie allerdings in Kauf nehmen, dass weder `stunnel` noch OpenSSH in der Lage sind, UDP-basierte Protokolle zu tunneln. Außerdem muss

der Tunnel immer vor dem eigentlichen Anwendungsprotokoll aufgebaut werden, was sich schon bei HTTPS störend bemerkbar macht. Protokolle, die dynamisch neue Verbindungen aushandeln, sind damit außen vor – ein prominentes Beispiel dieser Klasse ist FTP.

Dynamische Protokolle und solche, die (auch) UDP benutzen, müssen daher entweder auf Anwendungsebene abgesichert werden (wie FTPS, oder DNS mit TSIG), oder aber es wird gleich der gesamte Netzverkehr gesichert.

Netzwerkebene

Bei dieser Verschlüsselung auf Netzwerkebene stellt das System zusätzliche »virtuelle« Netzwerkschnittstellen zur Verfügung. Daten, die über diese Schnittstellen übertragen werden, werden verschlüsselt und über ein unsicheres Medium (etwa das Internet) an einen anderen Rechner übertragen, wo sie wiederum auf einer virtuellen Netzwerkschnittstelle ankommen. Das heißt, Sie können ohne weitere Konfiguration beliebige IP-basierte Protokolle übertragen. Man spricht von einem »virtuellen privaten Netz« (VPN).

Die beiden wichtigsten Anwendungsfälle für VPN sind die Anbindung von Heim- oder Außendienstarbeitern – auch *road warriors* genannt – an das interne Firmennetz und die Kopplung von lokalen Netzen an verschiedenen Standorten, so dass sie wie ein logisches Netz erscheinen.

Es gibt verschiedene Möglichkeiten, dies technisch anzugehen:

- Die simpelste Möglichkeit, VPN mit »Linux-Bordmitteln« zu implementieren, ist das Tunneln von PPP über eine OpenSSH-Verbindung (auch bekannt als »VPN für Arme«). Dieser Ansatz funktioniert nur zwischen Unix-Rechnern und hat gravierende Probleme, die ihn für die Praxis untauglich machen.

  Insbesondere gibt es Schwierigkeiten damit, wenn Sie in der PPP-Sitzung TCP-basierte Protokolle verwenden (was in der Praxis kaum zu vermeiden ist – schließlich machen Sie sich die Mühe überhaupt nur, weil Sie ja Sachen wie HTTP, POP3 und SMTP benutzen *wollen*). Unter dem Strich übertragen Sie dann TCP (über PPP) über TCP, und das führt zu Ineffizienzen und Fehlern, weil es passieren kann, dass die Steuerungsalgorithmen von TCP auf beiden Ebenen gegeneinander arbeiten. Genauer ist das in [Tit01] beschrieben.

- Das Microsoft-Protokoll PPTP (engl. *point-to-point tunneling protocol*) wird von Linux grundsätzlich unterstützt. Allerdings hat es Schwächen, die es nahelegen, PPTP nicht für sicherheitsrelevante Anwendungen zu verwenden [SMW99]. Machen Sie einen Bogen darum. PPTP

- IPsec ist eine umfassende Sicherheitsinfrastruktur, die für IPv6 definiert und auf IPv4 zurückportiert wurde. Infrastrukturanbieter wie Cisco unterstützen das, genau wie Linux. IPsec ist sehr komplex zu installieren und zu betreiben. IPsec

- OpenVPN verwendet TLS, um eine VPN-Infrastruktur zu realisieren. Es steht nicht nur für Linux, sondern auch für andere Plattformen (inklusive Windows) zur Verfügung. OpenVPN

Im Rest dieses Kapitels beschäftigen wir uns mit OpenVPN.

Übungen

9.1 [1] Erklären Sie den Unterschied zwischen SMTPS einerseits und ESMTP mit STARTTLS andererseits.

9.2 [2] Warum können Sie virtuelles Hosting, die Zusammenfassung von verschieden Web-Präsenzen auf einem Server, bei HTTPS-Servern nur IP-basiert durchführen (jede Präsenz benötigt eine eigene IP-Adresse), wohingegen HTTP-Server sich auch namensbasiert virtuell hosten lassen?

9.3 [2] Geben Sie die enzelnen Schichten an, die bei einem VPN durch PPP-über-SSH anfallen.

9.2 OpenVPN

9.2.1 Grundlagen

OpenVPN (`http://openvpn.net/`) ist eine frei verfügbare VPN-Infrastruktur auf der Basis von TLS, die sowohl die Anbindung einzelner Rechner an ein lokales Netz als auch die Kopplung lokaler Netze unterstützt. Sie können OpenVPN entweder mit »vorher ausgetauschten Schlüsseln« (*pre-shared keys*) betreiben oder die Authentisierung der beteiligten Rechner und Benutzer über X.509-Zertifikate sicherstellen. Ebenso können Sie OpenVPN im »Partner-Modus« (*peer mode*) betreiben, bei dem zwei gleichberechtigte OpenVPN-Installationen sich gegenseitig authentisieren, oder im »Server-Modus«, bei dem ein zentraler Server mehrere (oder viele) Verbindungen für externe Clients zur Verfügung stellt.

Wenn Sie aufgepasst haben und sich ein bisschen mit TLS auskennen, wissen Sie vielleicht, dass TLS eigentlich nur für zuverlässige Verbindungen wie TCP definiert ist, nicht für UDP. Wie schafft OpenVPN es dann, den kompletten Netzverkehr (TCP und UDP) zu übertragen, ohne einerseits UDP über TLS (und damit TCP) übertragen zu müssen und andererseits den oben erwähnten und in [Tit01] beschriebenen Problemen mit der Tunnelung von TCP über TCP zum Opfer zu fallen? Die Antwort darauf lautet, dass TLS zwar eine *zuverlässige* Verbindung benötigt, aber keineswegs auf »echtem« TCP besteht – TCP ist meistens nur die naheliegendste Wahl. OpenVPN definiert seine eigene zuverlässige Verbindungsschicht basierend auf UDP, die einerseits TLS das gibt, was es sehen möchte, aber andererseits so konstruiert ist, dass sie die TCP-über-TCP-Probleme vermeidet.

OpenVPN ist Bestandteil der gängigen Linux-Distributionen und darum leicht über deren Paketverwaltungswerkzeuge zu installieren. Alternativ finden Sie den Quellcode auf `http://openvpn.net/index.php/open-source/downloads.html` (dort stehen auch vorübersetzte Versionen für Microsoft Windows zur Verfügung).

Wenn Sie OpenVPN aus Quellcode übersetzen, sollten Sie sich die Mühe machen, die Authentizität und Integrität des Quellcodes zu prüfen. Instruktionen dafür stehen auf der Download-Seite.

9.2.2 Allgemeine Konfiguration

Neben der grundsätzlichen Betriebsart (Partner- oder Server-Modus) und der Art der Authentisierung (vorher ausgetauschte Schlüssel oder X.509-Zertifikate) müssen Sie sich überlegen, ob Sie OpenVPN auf der IP-Ebene (*routed mode*) oder der Ethernet-Ebene (*bridged mode*) betreiben wollen.

routed mode

- Der *routed mode* ist einfacher zu konfigurieren und in vielen Fällen effizienter. Er operiert auf der ISO/OSI-Schicht 3 und verwendet unter Linux ein Tunnel-Device (typischerweise `tun`*x* genannt), das es ermöglicht, IP-Datagramme über den verschlüsselten Transportweg zu leiten. Im gerouteten Modus bekommt die verschlüsselte Verbindung ein eigenes IP-Subnetz zugeordnet, und auf beiden beteiligten Rechnern werden Routen installiert, die dafür sorgen, dass der Datenverkehr zwischen ihnen über die verschlüsselte Verbindung (und nicht das unterliegende unverschlüsselte Internet) fließt.

bridged mode

- Auf der Ethernet-Ebene können Sie im *bridged mode* zwei lokale Ethernets transparent (auf der ISO/OSI-Schicht 2) vernetzen. Das heißt, ein Rechner,

der sich über OpenVPN mit einem entfernten Netz verbindet, bekommt eine IP-Adresse aus dem entfernten Netz und kann sofort auf sämtliche Stationen dort zugreifen. Bridging leitet auch Broadcasts weiter und funktioniert für Schicht-3-Protokolle außer IP. Die wesentlichen Nachteile sind, dass der Ansatz aufwendiger zu konfigurieren ist (typischerweise sind betriebssystemabhängige Werkzeuge nötig, um die von OpenVPN verwendete `tap`-Schnittstelle mit einer Ethernet-Schnittstelle der Station zu »bridgen«) und schlechter skaliert, also weniger gut mit einer großen Anzahl von Clients umgehen kann.

Der *routed mode* ist wesentlich gängiger, und Sie sollten ihn verwenden, wenn Sie nicht unbedingt die Eigenschaften des *bridged mode* wie die Weiterleitung von Ethernet-Broadcasts benötigen.

Linux stellt seit Kernel 2.4 (also für praktische Zwecke seit kurz nach der Eiszeit) die für Programme wie OpenVPN nötigen »virtuellen« Netzwerkschnittstellen zur Verfügung, die nicht mit Hardware-Treibern kommunizieren, sondern es möglich machen, die an sie gesandten Pakete weiterzubearbeiten und dann zum Beispiel an »echte« Netzwerkschnittstellen zu schicken. Der Vorteil ist, dass diese Schnittstellen (typischerweise `tun0`, `tun1`, … bzw. `tap0`, `tap1`, … genannt) aus der Sicht des Systems aussehen und funktionieren wie »echte« Netzwerkschnittstellen, das heißt, sie können mit Adressen und Routen versehen werden, Paketfilterregeln können sich auf sie beziehen und so weiter. Der Unterschied zwischen `tun`*x* und `tap`*y* ist, dass `tun`*x*-Schnittstellen mit IP-Datagrammen arbeiten (ISO/OSI-Schicht 3) und `tap`*y*-Schnittstellen mit Ethernet-Frames (ISO/OSI-Schicht 2).

virtuelle Netzwerkschnittstellen

Der nächste Schritt besteht darin, sich Gedanken über die Adressenvergabe zu machen. In aller Regel verwendet man für VPN-Konfigurationen Adressen aus den frei verfügbaren Bereichen nach [RFC1918], also aus den Netzen `10.0.0.0/8`, `172.16.0.0/12` sowie `192.168.0.0/16`. Es ist wichtig, die Adressbereiche für das VPN so zu wählen, dass sie weder mit anderweitig in Ihrer Installation (etwa für lokale Subnetze, die über NAT ans Internet angekoppelt sind) verwendeten Adressen kollidieren noch – vor allem für *road warriors* – mit Adressen, die wahrscheinlich von Hotels, Internet-Cafés und ähnlichen Betreibern öffentlicher Netzzugänge benutzt werden.

Adressenvergabe

Wenn Sie für Ihre VPN-Konfiguration zum Beispiel das Netz `192.168.0.0/24` benutzen, stehen die Chancen gut, dass ein Hotel oder Internet-Café Ihnen Ihre Adresse für das Internet aus demselben ziemlich naheliegenden Netz zuteilt. In diesem Moment haben Sie dann ein Problem.

Am geschicktesten verwenden Sie ein obskures Teilnetz mit einer `10` am Anfang, etwas wie `10.45.67.0/8`.

Wenn Sie mehrere unabhängige VPNs konfigurieren, sollten Sie es sich verkneifen, allen dieselben Adressbereiche zuzuteilen. Auch wenn das auf den ersten Blick einfach aussieht, kommt doch mit Sicherheit irgendwann der Tag, wo Rechner in diesen VPNs miteinander kommunizieren sollen, und dann ist das Heulen und Zähneklappern groß.

Für seinen Zugriff auf das Internet verwendet OpenVPN standardmäßig den Port `1194/udp`. Dieser Port muss natürlich auf dem äußeren Paketfilter freigeschaltet sein, wenn Sie eine Firewall-Infrastruktur verwenden, oder muss über eine Portweiterleitung vom externen Router auf den OpenVPN-»Server« umdirigiert werden.

Port

Viele Linux-Distributionen für Router (etwa OpenWRT) enthalten heute OpenVPN oder machen es leicht möglich, es nachzuinstallieren. Auch viele fertige Router auf Linux-Basis bringen OpenVPN mit. Während es ein gewisses Sicherheitsrisiko darstellt, den Router gleichzeitig als VPN-Gateway zu verwenden, ist das doch eine naheliegende Konfiguration, die für viele Zwecke ausreicht.

9.2.3 Einfache Tunnel

In diesem Abschnitt zeigen wir Ihnen einige simple Beispiele für OpenVPN im »Partner-Modus«:

ohne Verschlüsselung

Unverschlüsselte Daten Zu Testzwecken können Sie zwischen den Rechnern `red.example.com` und `blue.example.com` einen einfachen Tunnel ohne Verschlüsselung wie folgt aufbauen:

```
blue# openvpn --remote red --dev tun0 --ifconfig 10.45.67.1 10.45.67.2
red# openvpn --remote blue --dev tun0 --ifconfig 10.45.67.2 10.45.67.1
```

(beachten Sie, dass die beiden Adressen auf `red` in der umgekehrten Reihenfolge angegeben werden müssen wie auf `blue`). Anschließend sollten Sie über ein `ping` auf die entsprechende Adresse den anderen Rechner erreichen können:

```
blue# ping 10.45.67.2
PING 10.45.67.2 (10.45.67.2) 56(84) bytes of data.
64 bytes from 10.45.67.2: icmp_req=1 ttl=64 time=1.05 ms
◁◁◁◁◁
```

Sie können sich natürlich auch mit `ifconfig` und `route` davon überzeugen, dass alles korrekt aufgebaut wurde:

```
blue# route -n
Kernel-IP-Routentabelle
Ziel         Router         Genmask         Flags Metric Ref Use Iface
10.45.67.2   0.0.0.0        255.255.255.255 UH    0      0     0 tun0
192.168.61.0 0.0.0.0        255.255.255.0   U     0      0     0 eth0
0.0.0.0      192.168.61.254 0.0.0.0         UG    0      0     0 eth0
blue# ifconfig tun0
tun0      Link encap:UNSPEC  Hardware Adresse 00-00-00-00-00-00◁◁◁◁◁
          inet Adresse:10.45.67.1  P-z-P:10.45.67.2  ▷
           ◁ Maske:255.255.255.255
          UP PUNKTZUPUNKT RUNNING NOARP MULTICAST  MTU:1500  Metrik:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:100
          RX bytes:420 (420.0 B)  TX bytes:420 (420.0 B)
```

Wenn Sie dies über das Internet zum Laufen bekommen, dann wissen Sie zumindest, dass alle Firewall-Regeln stimmen. Das ist viel wert, aber nicht das, was wir eigentlich wollten – ohne Verschlüsselung und Authentisierung können Sie sich den Aufwand eigentlich sparen.

Schlüssel

Vorher ausgetauschtes Geheimnis Als nächstes können Sie versuchen, die beiden Rechner über einen *verschlüsselten* Tunnel zu verbinden. Im einfachsten Fall erreichen Sie das über ein »vorher ausgetauschtes Geheimnis« (*pre-shared key*). Das heißt, Sie erzeugen auf einem der beiden Rechner einen Schlüssel, ungefähr so:

```
# openvpn --genkey --secret /etc/openvpn/peer.key
```

Diesen Schlüssel übertragen Sie dann auf einem sicheren Transportweg (SSH, reitender Bote mit USB-Stick, …) auf den anderen Rechner und legen Sie ihn dort ebenfalls in `/etc/openvpn/peer.key` ab. Sie können den Tunnel dann mit einem Kommando wie

```
blue# openvpn --remote red --dev tun0 --ifconfig 10.45.67.1 10.45.67.2▷
 ◁  --secret /etc/openvpn/peer.key
```

starten (das korrespondierende Kommando für `red` überlassen wir Ihrer Vorstellungskraft).


Sie sollten natürlich darauf achten, dass die Schlüsseldatei nur für `root` lesbar ist.

Der `ping`-Test von oben sollte natürlich nach wie vor funktionieren. Ein kurzer Test mit einem Programm wie `wireshark` dürfte Sie jedoch davon überzeugen, dass die Daten jetzt verschlüsselt übertragen werden.


Die bequeme Konfiguration hat allerdings auch ihren Preis. Sollte der geheime Schlüssel jemals bekannt werden, so sind alle bisherigen mit diesem Schlüssel verschlüsselten Kommunikationsvorgänge entschlüsselbar. – Diese Gefahr der rückwirkenden Entschlüsselung können Sie dadurch eindämmen, dass Sie den geheimen Schlüssel regelmäßig austauschen, was allerdings wegen der Notwendigkeit eines unabhängigen sicheren Kanals dafür wiederum aufwendig sein kann.

Konfigurationsdatei Auf die Dauer ist es unbequem, OpenVPN mit einer langen Kommandozeile starten zu müssen, die alle benötigten Parameter enthält. Sie können die Kommandooptionen auch (ohne das »`--`« am Anfang) in eine Datei schreiben, etwa so:

```
blue# cat /etc/openvpn/peer.conf
# Konfigurationsdatei für blue.example.com

dev      tun0
remote   red
ifconfig 10.45.67.1 10.45.67.2
secret   /etc/openvpn/peer.key
```

Anschließend können Sie OpenVPN mit dem Kommando

```
blue# openvpn --config /etc/openvpn/peer.conf
```

starten.


Die OpenVPN-Pakete gängiger Linux-Distributionen enthalten meist Init-Skripte, die bei einem

```
# /etc/init.d/openvpn start
```

automatisch einen OpenVPN-Prozess für jede Datei in `/etc/openvpn` starten, deren Name mit »`.conf`« endet. Das ist sehr bequem. Meist können Sie auch mit etwas wie

```
# /etc/init.d/openvpn stop peer
```

auf einzelne OpenVPN-Prozesse einwirken.


Bei Debian GNU/Linux entscheidet die Variable `AUTOSTART` in der Datei `/etc/default/openvpn` darüber, ob (und wenn ja, welche) VPNs automatisch gestartet werden. Der Wert `all` steht für »alle«, `none` für »keine«, oder Sie können die gewünschten VPNs namentlich aufzählen. Außerdem können Sie den Start eines VPN an den Start der unterliegenden (unverschlüsselten) Netzwerkschnittstelle koppeln, indem Sie etwas wie

```
iface eth0 inet dhcp
  openvpn peer                                    Auch für VPN peer
```

nach `/etc/network/interfaces` schreiben. Ein

```
# ifup eth0
```

startet dann gleichzeitig auch OpenVPN mit der Konfigurationsdatei `/etc/openvpn/peer.conf`. Wenn Sie das VPN lieber manuell mit `ifup` und `ifdown` starten und stoppen wollen, können Sie statt dessen auch etwas wie

```
auto vpn
iface vpn inet manual
  openvpn peer
```

in `/etc/network/interfaces` ablegen. – Wenn Sie OpenVPN über `/etc/network/interfaces` starten, sollten Sie allerdings dafür sorgen, dass OpenVPN dieselben VPNs nicht auch über sein eigenes Init-Skript startet, indem Sie die Variable `AUTOSTART` in `/etc/default/openvpn` entsprechend setzen.

Übungen

9.4 [2] Schauen Sie nach, wie OpenVPN bei Ihrer Distribution funktioniert. Gibt es bequeme Steuerungsmöglichkeiten über das Init-Skript oder systemweite Voreinstellungen?

9.5 [!2] Bauen Sie wie in diesem Abschnitt gezeigt einen verschlüsselten Tunnen zwischen zwei Rechnern auf. Verwenden Sie dazu bequemerweise zwei verschiedene Rechner, zwei virtuelle Maschinen oder – in einem Präsenzkurs – verabreden Sie sich mit Ihrem Sitznachbarn. Überzeugen Sie sich mit einem Paketsniffer wie `tcpdump` oder `wireshark`, dass ein `ping` auf die in der OpenVPN-Konfiguration festgelegte Adresse am anderen Ende der verschlüsselten Strecke wirklich verschlüsselt übertragen wird und nicht im Klartext.

9.2.4 OpenVPN mit TLS und X.509-Zertifikaten

Statt mit gemeinsamen Schlüsseln zu arbeiten, können Sie auch X.509-Zertifikate einsetzen, um die »Partner« in einer OpenVPN-Infrastruktur zu authentisieren. Sie brauchen ein X.509-Zertifikat (und den dazugehörigen privaten Schlüssel) für jeden beteiligten Rechner. Außerdem brauchen Sie zur Authentisierung der Zertifikate das Zertifikat der Zertifizierungsstelle (CA), die die X.509-Zertifikate für die Partner ausgestellt hat.

Wir können an dieser Stelle nicht ausführlich auf die Generierung und Verwaltung von X.509-Zertifikaten eingehen. Anhang B gibt einen schnellen Überblick über die dafür nötigen Schritte. Mehr Informationen darüber finden Sie zum Beispiel in den Linup-Front-Schulungsunterlagen *Apache und SSL* oder *Linux als Web- und FTP-Server*.

Im Folgenden gehen wir davon aus, dass Sie über die folgenden Dateien verfügen:

`blue-cert.pem`	Zertifikat für den Rechner `blue`
`blue-key.pem`	Privater Schlüssel dazu
`red-cert.pem`	Zertifikat für den Rechner `red`
`red-key.pem`	Privater Schlüssel dazu
`ca-cert.pem`	Zertifikat der Zertifizierungsstelle

Die Konfiguration ist jetzt asymmetrisch, da für TLS einer der beiden Rechner »Server« und der andere »Client« sein muss. Dies hat nichts damit zu tun, ob Sie OpenVPN im Partner- oder im Server-Modus betreiben. In jedem Fall müssen sich beide Endpunkte authentisieren, egal wer der TLS-Client und wer der TLS-Server ist. In unserem Beispiel erklären wir `blue.example.com` zum TLS-Server:

```
# Konfiguration für blue (TLS-Server)

dev      tun0
remote   red
ifconfig 10.45.67.1 10.45.67.2

tls-server
dh    /etc/openvpn/dh2048.pem          Diffie-Hellman-Parameter (s.u.)
cert  /etc/openvpn/blue-cert.pem       Zertifikat für blue
key   /etc/openvpn/blue-key.pem        Privater Schlüssel dazu
ca    /etc/openvpn/ca-cert.pem         Zertifikat der CA
```

(Das Zertifikat der Zertifizierungsstelle, `ca-cert.pem`, braucht `blue`, um das Zertifikat von `red` zu überprüfen.) Die Schlüsseldatei sollte nur für `root` lesbar sein.

Die Datei `dh2048.pem` enthält »Diffie-Hellman-Parameter« zum sicheren Schlüsselaustausch – um sie zu erzeugen, brauchen Sie nur das Kommando

```
blue# openssl dhparam -out dh2048.pem 2048
```

und einige Geduld.

Hier ist die Konfiguration für `red.example.com`, den TLS-Client:

```
# Konfiguration für red (TLS-Client)

dev      tun0
remote   blue
ifconfig 10.45.67.2 10.45.67.1

tls-client
cert  /etc/openvpn/red-cert.pem        Zertifikat für red
key   /etc/openvpn/red-key.pem         Privater Schlüssel dazu
ca    /etc/openvpn/ca-cert.pem         Zertifikat der CA
```

Auf dem TLS-Client wird die Diffie-Hellman-Datei nicht gebraucht, aber das Zertifikat der Zertifizierungsstelle, `ca-cert.pem`, ist zur Überprüfung des Zertifikats von `blue` nötig.

Übungen

9.6 [!2] Setzen Sie die hier gezeigte OpenVPN-Konfiguration mit TLS um. Was passiert, wenn ein privater Schlüssel mit einer Passphrase versehen ist?

9.2.5 Server-Modus

Im Server-Modus kann ein einziger OpenVPN-Server – mit einem einzigen lokalen Port – Hunderte oder Tausende von Clients bedienen. (Natürlich wollen Sie dafür dann nicht die möhrigste Hardware verwenden.) Er eignet sich damit hervorragend für die Anbindung vieler *road warriors* mit minimalem Konfigurationsaufwand. Für den Server-Modus ist TLS-basierte Verschlüsselung mit X.509-Zertifikaten Pflicht.

Server-Konfiguration Um einen Rechner als OpenVPN-Server zu konfigurieren, sind einige weitere Einstellungen in der Konfigurationsdatei nötig:

```
mode server
server 10.46.78.0 255.255.255.0
keepalive 10 60
```

Mit »mode server« wird der Rechner in den Server-Modus versetzt. Die server-Direktive gibt einen Adressenbereich an, aus dem Adressen an die Clients vergeben werden. Dabei behält der Server immer die erste Adresse (hier 10.46.78.1) für sich.

Wenn Sie nichts Anderes einstellen, dann weist der OpenVPN-Server jedem Client ein /30-Subnetz zu. Das heißt, der erste Client bekommt in unserem Beispiel die Adresse 10.46.78.6 und der OpenVPN-Server gibt sich selbst die Adresse 10.46.78.5, beide aus dem Netz 10.46.78.4/30. Dies ist erforderlich, damit OpenVPN beliebige Windows-Clients bedienen kann, aber impliziert, dass Sie mit einem /24-Adressblock in der server-Direktive »nur« maximal 63 Clients gleichzeitig bedienen können. Konfigurieren Sie also gegebenenfalls einen größeren Adressblock (es ist ja in der Regel nicht so, dass es nicht genug Adressen gäbe).

Sollte Ihnen diese Methode ein Dorn im Auge sein und Sie es nicht mit älteren Windows-Versionen als Clients zu tun bekommen, können Sie auf dem Server die Direktive

```
topology subnet
```

setzen. In diesem Fall bekommen alle Clients fortlaufende Adressen aus dem mit server angegebenen Netz. Ein Nebeneffekt ist jedoch, dass die Clients sich gegenseitig sehen können, was bei der Standardtopologie (die Sie übrigens explizit über

```
topology net30
```

einstellen können) nicht der Fall ist. Möglicherweise ist das ein Sicherheitsproblem.

Client-Konfiguration Auf einem Rechner, der als Client an einem OpenVPN-Server fungieren soll, müssen Sie in der Konfigurationsdatei die Direktive

```
client
```

setzen. Damit ändert die remote-Direktive ihre Bedeutung; sie gibt den Server an, mit dem der Client sich in Verbindung setzen möchte.

mehrere remote-Direktiven

Sie können auch mehrere remote-Direktiven angeben, die auf verschiedene OpenVPN-Server verweisen. In diesem Fall versucht der Client, sich mit dem erstgenannten Server zu verbinden; gelingt das nicht, wird der zweite ausprobiert und so weiter. Auf diese Weise können Sie ein fehlertolerantes System aufbauen.

Plausibilität

Aus Sicherheitsgründen ist es sinnvoll, dass der Client das Server-Zertifikat nicht nur formal auf Richtigkeit prüft (ob es mit dem lokal verfügbaren Wurzelzertifikat signiert wurde), sondern auch auf Plausibilität. Ziel ist es, eine Situation zu vermeiden, in der ein anderer Client sich als OpenVPN-Server ausgibt. Dafür gibt es verschiedene Möglichkeiten:

- Mit der Direktive tls-remote können Sie OpenVPN anweisen, nur Verbindungen von anderen Rechnern anzunehmen, deren »Common Names« im Zertifikat auf den dort angegebenen Namen passt. Ein

  ```
  tls-remote blue.example.com
  ```

  in der Konfiguration des Clients erlaubt also nur Verbindungen mit einem Rechner, der ein Zertifikat vorweisen kann, das auf blue.example.com ausgestellt ist.

- Mit `tls-verify` können Sie ein Skript aufrufen, das auch noch andere Eigenschaften des Zertifikats überprüft. Details darüber stehen in `openvpn`(8).
- Sie können beim Erzeugen des Zertifikats arrangieren, dass dieses mit bestimmten Eigenschaften ausgestattet wird, namentlich (für ein Server-Zertifikat)

  ```
  nsCertType = server
  keyUsage = digitalSignature, keyEncipherment
  extendedKeyUsage = serverAuth
  ```

  Anschließend fügen Sie zur Client-Konfiguration ein

  ```
  remote-cert-tls server
  ```

  hinzu.
- Sie können Client-Zertifikate mit einer anderen (Sub?)-CA signieren als Server-Zertifikate. In diesem Fall müssen Sie auf den Clients das Zertifikat der Server-CA als `ca` installieren und auf den Servern das Zertifikat der Client-CA.

Netze koppeln Im gerouteten Modus etabliert OpenVPN auf dem Client eine Route zum Server, aber nicht notwendigerweise darüber hinaus. Wenn Sie andere Stationen im Netz des Servers zugänglich machen wollen, müssen Sie dafür sorgen, dass der Server dem Client eine passende Route schickt, die Pakete für das Netz »hinter« dem Server über die verschlüsselte Verbindung leitet. (Es kann ja auch sein, dass dieses Netz gar nicht auf dem direkten Weg zugänglich ist, etwa weil es sich um ein firmeninternes LAN mit einer RFC-1918-Adresse handelt.) Sie erreichen das mit einer Direktive der Form

Netz des Servers

```
push "route 192.168.62.0 255.255.255.0 vpn_gateway"
```

in der Konfiguration auf dem *Server*. Diese Route wird dann beim Verbindungsaufbau an den Client gesendet und dort in Kraft gesetzt.

Das `vpn_gateway` in der Konfiguration steht für den VPN-Server, ohne dass man dessen Adresse explizit in die Route schreiben muss.

Damit diese Anbindung funktioniert, müssen Sie auf dem OpenVPN-Server IP-Forwarding einschalten. Das geht mit

```
# echo 1 >/proc/sys/net/ipv4/ip_forward
```

bzw. einer distributionsabhängigen Methode für die dauerhafte Aktivierung.

Wenn Sie auch auf der Client-Seite ein LAN anbinden wollen, so dass mehrere Rechner über die verschlüsselte Verbindung kommunizieren können, müssen Sie noch etwas mehr arbeiten. Zuerst müssen Sie auch auf dem OpenVPN-Client IP-Forwarding in Kraft setzen. Außerdem müssen Sie darauf achten, dass es keine Kollision zwischen dem Adressbereich des LANs auf der Client-Seite und anderen Netzen »in der Nähe« gibt. Insbesondere sollte auch auf anderen Clients nicht derselbe Adressbereich für ein LAN benutzt werden, weil das das Routing auf dem OpenVPN-Server durcheinanderbringt.

Clientseitiges LAN

Auf der Server-Seite brauchen Sie ein Verzeichnis für »clientspezifische Konfiguration« – etwa `/etc/openvpn/ccd` –, auf das Sie in der Konfiguration des Servers mit

```
client-config-dir /etc/openvpn/ccd
```

verweisen. Darin können Sie Dateien ablegen, die so heißen wie die »Common Names« von Clients (also zum Beispiel `red.example.com`) und die Konfigurationseinstellungen enthalten, die dann nur für den betreffenden Client gelten. In unserem Beispiel könnte in der Datei `/etc/openvpn/ccd/red.example.com` etwas stehen wie

```
iroute 192.168.111.0 255.255.255.0
```

wenn das Netz `192.168.111.0/24` über `red` angebunden werden soll. Außerdem brauchen Sie noch in der Konfiguration des Servers (nicht der clientspezifischen Datei!) die Zeile

```
route 192.168.111.0 255.255.255.0
```

Auch wenn das aussieht wie eine unnötige und unverständliche Verdopplung, ist es doch keine: Die `route`-Direktive in der allgemeinen Serverkonfiguration sorgt dafür, dass im Linux-Kern eine Route für das Client-Netz gesetzt wird, die die betreffenden Daten an OpenVPN leitet. Die `iroute`-Direktive (kurz für *internal route*) hingegen ist zuständig für die Zuordnung innerhalb von OpenVPN.

Damit müßte eigentlich alles erledigt sein, und die Rechner in beiden lokalen Netzen sollten einander erreichen können.

Übungen

9.7 [!2] Konfigurieren Sie Ihre beiden Rechner so, dass der eine als OpenVPN-Server und der andere als OpenVPN-Client agiert. Welche IP-Adressen bekommen die Rechner? Welche Routen werden auf dem Server und dem Client gesetzt, wenn die Verbindung aktiv ist?

9.8 [2] Testen Sie, was passiert, wenn ein Client sich mit einem X.509-Zertifikat anmeldet, das gleichzeitig schon von einem anderen Client für eine Sitzung verwendet wird. Was können Sie tun, um dieses Verhalten zu unterdrücken?

9.9 [3] (Großes Projekt.) Probieren Sie die Kopplung zweier lokaler Netze im gerouteten Modus von OpenVPN aus. Sie brauchen dafür zwei zusätzliche Rechner (spätestens hier fängt Virtualisierung an, sich zu lohnen), von denen der eine – nennen wir ihn im Beispiel `cyan.example.com` – mit `blue.example.com` und der andere – nennen wir ihn `pink.example.com` – mit `red.example.com` verbunden ist. (`blue` und `red` brauchen dafür jeweils zwei Netzwerkkarten.) Zwischen `cyan` und `pink` sollte keine direkte Verbindung bestehen. – Konfigurieren Sie zuerst das serverseitige LAN zwischen `blue` und `cyan` und vergewissern Sie sich, dass Sie von `red` aus den Rechner `cyan` erreichen können und die entsprechenden Datenpakete über die verschlüsselte Verbindung laufen. Konfigurieren Sie dann das LAN auf der Client-Seite und stellen Sie sicher, dass der Rechner `pink` den Rechner `cyan` erreichen kann und dass auch hier die verschlüsselte Verbindung benutzt wird.

Zusammenfassung

- Der Datenverkehr in einem Netz kann auf Anwendungs-, Transport- oder Netzwerkebene abgesichert werden.
- Virtuelle private Netze (VPN) dienen zur sicheren Verbindung zweier oder mehrerer Rechner oder Netze über ein unsicheres Transportnetz (etwa das Internet).
- IPsec ist ein standardisiertes Verfahren zur Implementierung von VPNs (unter anderem), aber relativ kompliziert zu installieren und zu betreiben.
- OpenVPN ist eine einfache, portable VPN-Lösung auf der Basis von TLS.
- OpenVPN erlaubt die Kopplung von Rechnern und Netzen auf den Ebenen der ISO/OSI-Schichten 2 und 3 und die Authentisierung über vorher ausgetauschte Schlüssel oder X.509-Zertifikate.
- OpenVPN unterstützt auch einen flexiblen Server-Modus zur Anbindung von Hunderten oder Tausenden externer Rechner.

Literaturverzeichnis

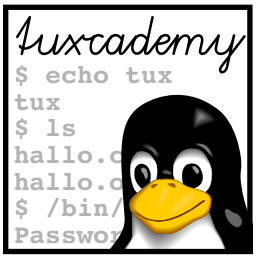
Gut04 Peter Gutmann. »Schutz (be)dürftig. Gravierende Fehler in VPN-Protokollen und deren Lösungen«. *Linux Magazin*, Januar 2004. S. 84–93.

RFC1918 Y. Rekhter, B. Moskowitz, D. Karrenberg, et al. »Address Allocation for Private Internets«, Februar 1996. `http://www.ietf.org/rfc/rfc1918.txt`

SMW99 Bruce Schneier, Mudge, David Wagner. »Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)«. Baumgart [**?**] S. 192–203.

Tit01 Olaf Titz. »Why TCP over TCP is a bad idea«, April 2001. `http://sites.inka.de/sites/bigred/devel/tcp-tcp.html`

Zel08 Thomas Zeller. *OpenVPN kompakt*. Brain-Media.de, 2008. ISBN 978-393931651-0.



A

Musterlösungen

Dieser Anhang enthält Musterlösungen für ausgewählte Aufgaben.

1.2 Siehe hierzu z. B. S. 78.

2.1 Grundsätzlich kann man sich natürlich auf den Standpunkt stellen, dass ein Web-Server keinen Secure-Shell-Zugriff braucht. In der Praxis müssen Sie aber irgendwie Inhalte auf den Server kriegen, und auch für verschiedene Administrationsaufgaben ist der Shellzugriff nützlich. Wir würden Ihnen also durchaus dazu raten, auf dieses wichtige Stück Infrastruktur nicht zu verzichten. Selbst wenn der Web-Server nicht im klimatisierten Rechnerraum steht, sondern neben Ihrem Schreibtisch-PC, und Sie Monitor und Tastatur vor sich haben, ist ein Secure-Shell-Zugang etwa für automatische Sicherheitskopien sehr empfehlenswert. Achten Sie darauf, dass keine Unbefugten sich anmelden können – etwa indem Sie die Anzahl der Benutzeraccounts minimieren – und bestehen Sie auf einer Authentisierung über asymmetrische Kryptographie, so dass nur diejenigen Benutzer Zugriff erhalten, deren öffentliche Schlüssel auf dem Server abgelegt sind. Verbieten Sie direktes Anmelden als `root` und konfigurieren Sie den Paketfilter des Rechners so, dass nur designierte Management-Rechner Zugriff auf den `ssh`-Port haben. Damit müßten Sie die meisten Problemquellen ausgeschaltet haben.

3.2 »`PermitRootLogin yes`« legt fest, dass `root` nicht von vornherein ausgesperrt wird, aber sagt nichts über andere Benutzer. Mit »`AllowUsers root`« darf sich *nur* `root` anmelden – jedenfalls sofern es keine anderen `AllowUsers`- oder `AllowGroups`-Direktiven gibt.

3.4 Die größte Gefahr besteht darin, ein schwaches Kennwort für `root` zu verwenden. Wenn ein solcher Rechner direkt mit dem Internet verbunden ist, könnte ein Cracker, der systematisch nach Rechnern mit SSH-Zugängen für `root` sucht (und solche niederen Lebensformen gibt es leider wirklich zuhauf), sich unbefugten Zugang zum Rechner verschaffen. Für Benutzerkonten, die nicht `root` heißen, ist die Gefahr geringer, weil ein Angreifer erst den Namen des Kontos raten muss, was Sie natürlich nicht als Entschuldigung dafür verstehen sollten, ein schwaches Kennwort zu benutzen.

Wenn `root` sich überhaupt nicht direkt anmelden darf, bleibt zunächst, wie beschrieben, der Umweg über ein normales Benutzerkonto, gefolgt von `su` oder `sudo`. Sie sollten aber auf jeden Fall erwägen, überhaupt keine kennwortbasierten Zugriffe zuzulassen, sondern nur die Authentisierung über Schlüsselpaare zu verwenden und in der `sshd`-Konfigurationsdatei

```
PasswordAuthentication no
PubkeyAuthentication yes
```

zu setzen. Alternativ sollten Sie zumindest

```
PermitRootLogin without-password
```

in Erwägung ziehen.

3.5 Probieren Sie

```
# echo "Systemwartung ab jetzt für 1 Stunde" >/etc/nologin
# echo "rm /etc/nologin" | at now + 1 hours
```

3.7 Die zweite Sitzung bleibt davon völlig unbeeinflusst (zum Glück).

3.9 Pro öffentlichem Schlüssel können Sie leider nur eine `command`-Direktive angeben. Allerdings können Sie ein Programm schreiben, das Sie mit `command` ausführen und das das vom Benutzer angegebene Kommando prüft, das der Client in der Umgebungsvariable `SSH_ORIGINAL_COMMAND` übergeben hat. Ist das ursprünglich angegebene Kommando akzeptabel, wird es danach ausgeführt, ansonsten wird eine Fehlermeldung zurückgegeben.

3.11 Hier ist ein mögliches Beispiel:

```
#!/bin/bash
# sign-ssh-key PUBKEYFILE

pubkey=$1
cakey=${CAKEY:-$HOME/.ssh/ca-key}
serialfile=${CASERIAL:-$HOME/.ssh/.serial}
certsfile=${CACERTS:-$HOME/.ssh/certs-issued}

read -p "Key ID: " keyid
read -p "Principal: " principal

[ -f $serialfile ] || echo 0 >$serialfile
serial=$(cat $serialfile)

echo >&2 "Signing key (needs passphrase for $cakey)"
if ssh-keygen -s $cakey -I "$keyid" -n "$principal" \
   -V +52w -z $serial $pubkey
then
   echo $serial:$keyid:$principal >>$certsfile
   echo $((serial+1)) >$serialfile
else
   echo >&2 "Error signing key, no certificate issued"
   exit 1
fi
```

Beachten Sie die Anführungszeichen im `ssh-keygen`-Aufruf, die nötig sind, damit Leerzeichen in der Schlüsselkennung (und möglicherweise dem Prinzipal) die Kommandozeile nicht durcheinander bringen. Beachten Sie ferner, dass die Seriennummer in der Datei `.ssh/.serial` nur inkrementiert wird, wenn das Zertifikat erfolgreich erstellt werden konnte. Die Dateinamen für den Schlüssel der Zertifizierungsstelle, die Protokoll- und die Seriennummerdatei können Sie auch über Umgebungsvariable konfigurieren.

3.14 Mit einem Zertifikat ohne Prinzipal können Sie sich nicht anmelden, wenn der öffentliche Schlüssel der Zertifizierungsstelle systemweit (und nicht im angestrebten Benutzerkonto) installiert ist. Gäbe es diese Einschränkung nicht, könnten Sie mit Ihrem Zertifikat die Identität jedes beliebigen Benutzers auf dem entfernten System annehmen.

3.15 Die Optionen werden mit dem öffentlichen Schlüssel mit signiert und können darum nicht geändert werden, ohne das Zertifikat ungültig zu machen. Sie sind also nicht unter der Kontrolle des Zertifikat-Inhabers, sondern unter der des Zertifikat-Ausstellers – etwa so wie die `authorized_keys`-Dateien, wo diese Optionen sonst stehen würden.

4.1 Ein nur auf Schicht 2-Ebene arbeitender Paketfilter wäre nicht besonders sinnvoll, da er lediglich die MAC-Adressen von Ethernet-Frames als Filterkriterium heranziehen könnte. Müßten Sie sich auf Schicht 3 beschränken, könnten Sie zwar IP-Adressen zum Filtern verwenden, aber TCP- und UDP-Ports sowie TCP-Flags als Schicht-4-Artefakte stehen Ihnen auch da nicht zur Verfügung, und das wäre ein herber Verlust.

4.3 Verwenden Sie Fragmentierung. Viele Systeme prüf(t)en nicht, ob die Startadresse eines Fragments im ursprünglichen Datagramm plus die Fragmentlänge kleiner ist als 65536.

5.1 Wenn Sie kein Modul namens `iptables_filter.ko` im Verzeichnisbaum Ihres Kernels unter `/lib/modules` finden, könnte es sein, dass die grundlegende Funktionalität statisch in den Kernel eingebaut wurde. Natürlich können Sie aus der Existenz diverser dynamisch ladbarer Erweiterungsmodule auf Ihrem System schließen, dass Ihr Kernel Netfilter unterstützt. Sie können aber auch schauen, ob Ihre Distribution die bei der Erstellung Ihres Kernels verwendete Konfigurationsdatei mitliefert (typischerweise in `/boot/config-*`) und ob dort die Variable `CONFIG_NETFILTER` den Wert »y« hat.

5.2 Innerhalb von Netfilter-Regelsätzen können Sie sich durch Variable, Fallunterscheidungen, Schleifen und ähnliches viel Schreibarbeit sparen. Die Netfilter-Entwickler haben es vorgezogen, die Programmiermöglichkeiten der Shell »wiederzuverwenden«, statt eine eigene Spezifikationssprache für Netfilter zu entwickeln, die diese Merkmale auch unterstützt. Da Regelsätze zumeist einmal installiert und dann nicht ständig geändert werden, ist der Zusatzaufwand durch das wiederholte Starten von `iptables` im wesentlichen vernachlässigbar.

5.3 Zum Beispiel:

1. `-p icmp --icmp-type echo-request`
2. `-p tcp --dport 22 --syn`
3. `-p tcp -d 10.0.0.0/8 --dport 80`

5.4 Ohne das `state`-Modul wäre es nötig, für jedes durchzulassende Protokoll separate Regeln für eingehende und ausgehende Pakete aufzustellen. Mit `state` werden Pakete, die eine bestehende Verbindung fortsetzen, automatisch identifiziert, so dass die Anzahl der benötigten Regeln im wesentlichen halbiert wird. Dies macht die Regelsätze übersichtlicher und verringert den Wartungsaufwand beträchtlich.

5.5 Natürlich könnten Sie vor jeder DROP-Regel eine LOG-Regel mit denselben Auswahlkriterien plazieren. Das ist aber ineffizient und unangenehm zu warten. Definieren Sie lieber eine neue Kette:

```
iptables -N logdrop
iptables -A logdrop -j LOG
iptables -A logdrop -j DROP
```

Anschließend können Sie überall da, wo Sie bisher »`-j DROP`« gesagt haben, »`-j logdrop`« verwenden.

5.6 Vor allem dann, wenn das zu beantwortende Paket schon eine ICMP-Fehlermeldung ist. In diesem Fall hält man besser schön still, damit es nicht zu einem endlosen Ping-Pong-Spiel kommt. Es gibt noch ein paar andere Fälle, siehe [Packet-Filtering-HOWTO].

5.8 Wie Sie in Übung 5.7 feststellen konnten, entspricht ein DROP nicht dem »natürlichen« Verhalten eines ungefilterten Systems, wodurch einem Angreifer die Existens des Filters verraten wird.

Die REJECT-Regeln spielen die Antwort des Kernels nach, so als wenn nicht gefiltert würde. Da die »natürliche« Antwort protokollspezifisch ist, sind drei verschiedene Regeln erforderlich.

5.9 Wir beschränken uns auf die Unterschiede zum Praxisbeispiel in Abschnitt 5.3.6. Zunächst bemerken wir, dass die externe Adresse vom Provider dynamisch vergeben wird; EXT_IP ist demnach nicht definiert.

Nach der Installation der kernelbasierten Sicherheitsmaßnahmen in /proc können wir als erstes das Masquerading in Kraft setzen:

```
# einfaches Masquerading nach extern aktivieren
iptables -t nat -A POSTROUTING -o $EXT_IF -j MASQUERADE
```

Im Gegensatz zum Beispiel in Abschnitt 5.3.6 fungiert der Rechner jetzt als Router für Pakete, das heißt, wir müssen IP-Forwarding im Kernel aktivieren und auch in der FORWARD-Kette Antwortpakete durchleiten:

```
echo 1 >/proc/sys/net/ipv4/ip_forward

 -t filter -A FORWARD -m state --state ESTABLISHED,RELATED \
    -j ACCEPT
```

Neben Pings auf den Router erlauben wir auch Pings ins Internet:

```
# ping von innen auf Firewall erlauben
 -t filter -A INPUT -s $LOC_NET -d $LOC_IP \
    -p icmp --icmp-type ping -i $LOC_IF -j ACCEPT

on innen nach extern erlauben
 -t filter -A FORWARD -s $LOC_NET -d $EXT_NET \
    -p icmp --icmp-type ping -i $LOC_IF -o $EXT_IF -j ACCEPT
```

Rechner aus dem internen Netz bekommen Zugriff zum DNS-Server auf dem Router:

```
# DNS-Anfragen von innen zum lokalen DNS-Server erlauben
 -t filter -A INPUT -s $LOC_NET -d $LOC_IP \
    -p udp --sport 1024: --dport 53 -i $LOC_IF \
    -m state --state NEW -j ACCEPT
```

```
 -t filter -A INPUT -s $LOC_NET -d $LOC_IP \
    -p tcp --sport 1024: --dport 53 -i $LOC_IF \
    -m state --state NEW -j ACCEPT

fragen nach extern zum $DNSSERVER erlauben
 -t filter -A OUTPUT -d $DNSSERVER \
    -p udp --sport 1024: --dport 53 -o $EXT_IF \
    -m state --state NEW -j ACCEPT
 -t filter -A OUTPUT -d $DNSSERVER \
    -p tcp --sport 1024: --dport 53 -o $EXT_IF \
    -m state --state NEW -j ACCEPT
```

7.1 Ein Angreifer könnte z. B. in `/tmp` eine bösartige Version von `ls` ablegen. Sollte jemand innerhalb von `/tmp` sich mittels `ls` orientieren wollen, so wird er die modifizierte Version starten. – Das funktioniert auch dann noch, wenn »`.`« am Ende des Suchpfades liegt: das bösartige Programm muss dann nur `ls-l` heißen. Der Angreifer hofft darauf, dass sich ein Administrator irgendwann vertippt.

7.2 Wie in Übung 7.1 zu sehen ist, geht eine Gefahr auch von *hinzugefügten* Programmen aus.

7.3 Nur einige Vorschläge: `ls`, `ps`, `netstat`, `lsof`, `lsmod`, `iptables`, das Kernel-Image (`vmlinuz`).

7.4 Schreibgeschützte Datenbanken sind umständlicher zu aktualisieren als signierte Datenbanken. Dafür kann ein Angreifer eine signierte Datenbank einfach löschen; Sie wissen dann zwar, dass etwas nicht stimmt, aber um das aufwändige Neuaufsetzen des Systems kommen Sie trotzdem nicht herum.

7.5 Ein Angreifer könnte das Programm `tripwire` manipulieren, so dass es nicht mehr Alarm schlägt. Oder der Angreifer baut seine Hintertür nur ins *laufende* System ein und verzichtet darauf, diese im Dateisystem abzulegen (beliebt hierfür: selbstversteckende Kernel-Module).

Wenn Sie sicher gehen wollen, müssen Sie daher ein System letztendlich immer *von außen* überprüfen, d. h. mittels Rettungssystem, wobei Tripwire-Programm und Datenbank ebenfalls mitgebracht werden müssen.

7.8 Die Variable `IgnoreNone` enthält sowohl die Tests `CHMS` für den Dateiinhalt als auch den Test `a` für die Zugriffszeit. Die Berechnung der Prüfsummen erfordert aber, dass Tripwire den Dateiinhalt *liest* – folglich verändert die Überprüfung selbst die Zugriffszeit. Tripwire schlägt somit *immer* Alarm.

7.9 Ein Angreifer kann beispielsweise mit `touch` die Zugriffszeit und die Änderungszeit verstellen. Dabei wird jedoch immer auch die Metadaten-Änderungszeit mit gesetzt. Für deren direkte Manipulation gibt es aber keinen Systemaufruf; um diese Zeit zu verändern müsste entweder direkt auf die Festplatte geschrieben oder der Kernel manipuliert werden.

7.11 `shadow`, `mtab`, `passwd`, `resolv.conf` (bei DHCP), `lvm*` (Scan von LVM beim Booten), um nur einige zu nennen.

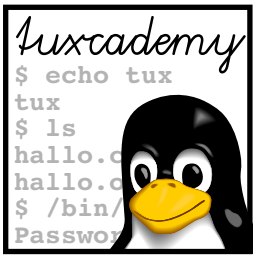
7.12 Neben den Heimatverzeichnissen `/home/*` und `/root` die Temporär-Verzeichnisse `/tmp` und `/var/tmp`, die Warteschlangen-Verzeichnisse unterhalb von `/var/spool`, die Mail-Boxen `/var/mail` und diverse Zustandsverzeichnisse (`/var/cache`, `/var/lock`, `/var/run`). Wenn sie `logrotate` o. Ä. benutzen auch `/var/log`.

9.1 Bei SMTPS wird ein eigener Port (`465/tcp`) benutzt, über den ein sicherer TLS-Tunnel aufgebaut wird; *danach* sprechen Client und Server normales SMTP. Bei der anderen Variante verbindet sich der Client auf Port `25/tcp` und Client und Server sprechen ESMTP. Bietet der Server das Kommando STARTTLS an, so kann der Client auf Verschlüsselung umschalten.

9.2 Beim namensbasierten virtuellen Hosting erkennt der Server an der `Host:`-Kopfzeile, mit welcher Web-Präsenz Sie kommunizieren wollen. Da HTTPS jedoch HTTP in einem SSL/TLS-Tunnel ist, kann der Server diese Zeile nicht lesen, bevor der Tunnel aufgebaut ist. Damit weiß der Server aber nicht, welches Server-Zertifikat er dem Client präsentieren muss. Ein Zertifikat mit dem falschen Rechner-Namen führt aber (hoffentlich!) zu einer Ablehnung durch den Client. (Erweiterungen für TLS, mit denen ein Client beim Verbindungsaufbau sagen kann, mit welchem aus einer Anzahl von Namen eines Servers er tatsächlich reden möchte, sind in Arbeit.)

9.3 Ethernet – IP – TCP – SSH – PPP – IP – TCP/UDP – Anwendung

9.8 Wenn ein neuer Client sich mit einem Zertifikat anmeldet, dessen »Common Name« dem eines anderen gerade aktiven Clients entspricht, wird die existierende Sitzung abgebrochen. Sie können das verhindern, indem Sie in der Konfiguration des Servers die Direktive `duplicate-cn` hinzufügen, die erlaubt, dass unter dem gleichen »Common Name« mehrere Sitzungen gleichzeitig stattfinden dürfen.



B

X.509-Crashkurs

Dieser Anhang gibt einen schnellen Überblick über X.509 und die Generierung von Zertifikaten. Wenn Sie genau wissen wollen, wie das alles funktioniert, lesen Sie die Linup-Front-Schulungsunterlagen *Apache und SSL* (APW2) oder *Linux als Web- und FTP-Server* (WEBF).

B.1 Einleitung: Kryptografie, Zertifikate und X.509

symmetrische Kryptografie

Die verschlüsselte Übertragung von Daten ist grundsätzlich ein gelöstes Problem: Symmetrische Kryptoverfahren wie AES machen es möglich, große Datenmengen effizient und für alle praktischen Zwecke unknackbar zu übertragen. Das Problem bei diesen Verfahren ist lediglich der Schlüsselaustausch: Da beide Kommunikationspartner über denselben geheimen Schlüssel verfügen müssen, müssen sie sich irgendwie vertraulich auf einen Schlüssel einigen können.

asymmetrische Kryptografie

Dieses Dilemma löst man über asymmetrische Kryptografie. Bei asymmetrischen Kryptoverfahren wie RSA hat jeder Teilnehmer zwei Schlüssel, einen privaten (geheimen) und einen öffentlichen. Der öffentliche Schlüssel kann allgemein bekannt gemacht werden, solange der private Schlüssel vertraulich bleibt. Dadurch wird es möglich, zwei verschiedene Probleme zu adressieren:

Verschlüsselung Alice[1] möchte eine vertrauliche Nachricht an Bob schicken. Sie verschafft sich Bobs öffentlichen Schlüssel und verschlüsselt damit die Nachricht. Bob verwendet seinen privaten Schlüssel, um die Nachricht wieder lesbar zu machen.

Digitale Signatur Alice möchte kundtun, dass sie ein bestimmtes Dokument verfasst hat. Sie »signiert« (verschlüsselt) das Dokument mit ihrem privaten Schlüssel. Bob (oder wer auch immer sonst über Alices öffentlichen Schlüssel verfügt) kann mit Alices öffentlichem Schlüssel verifizieren, dass Alice (und nicht sonst jemand) das Dokument signiert hat und dass der Inhalt authentisch ist, also dem entspricht, was Alice geschrieben hat.

Probleme mit asymmetrischer Kryptografie

Ein praktisches Problem ist, dass man mit asymmetrischen Kryptoverfahren nur relativ kleine Datenmengen verschlüsseln kann und dass sie relativ ineffizient sind – symmetrische Kryptoverfahren beruhen im Wesentlichen auf Bitoperationen, die sich sogar in Hardware realisieren lassen, während asymmetrischen Kryptoverfahren aufwendige mathematische Operationen zugrunde liegen. (RSA zum Beispiel rechnet mit sehr langen Zahlen herum.) In der Praxis verwendet man

[1] In der kryptografischen Literatur sind die beiden Kommunikationspartner traditionell immer »Alice« und »Bob«.

asymmetrische Kryptografie deshalb zusammen mit anderen kryptografischen Verfahren:

Sitzungsschlüssel **Verschlüsselung** Alice wählt einen zufälligen Schlüssel (den »Sitzungsschlüssel« oder *session key*) für ein geeignetes symmetrisches Kryptoverfahren (etwa AES). Sie verschlüsselt diesen Schlüssel mit Bobs öffentlichem Schlüssel und schickt das Resultat an Bob. Bob entschlüsselt den AES-Schlüssel mit seinem privaten Schlüssel. Anschließend können Alice und Bob vertraulich kommunizieren, indem sie ihre Nachrichten mit AES verschlüsseln.

Prüfsumme **Digitale Signatur** Alice bildet eine »kryptografische Prüfsumme« über das Dokument, also eine unumkehrbare Funktion, die von jedem Bit der Eingabe abhängt (Stichwort: MD5, SHA-1 und Ähnliches). Anschließend signiert sie diese Prüfsumme mit ihrem privaten Schlüssel. Zur Prüfung der Signatur bildet Bob die kryptografische Prüfsumme über das Dokument. Wenn die Entschlüsselung der Signatur mit Alices öffentlichem Schlüssel (den Bob ja hat) dasselbe Ergebnis liefert, ist die Signatur gültig, und das Dokument ist authentisch.

Auf diese Weise wird das Problem der Schlüsselverteilung gelöst, aber an seine Stelle tritt ein neues Problem: Angenommen, Bob findet irgendwo im Netz Alices öffentlichen Schlüssel (oder einen öffentlichen Schlüssel, der behauptet, zu Alice zu gehören). Wie kann Bob sicher sein, dass dieser Schlüssel *wirklich* Alices Schlüssel ist?

Eine mögliche Lösung für dieses Problem besteht darin, den Zusammenhang zwischen Alice und Alices Schlüssel von einer vertrauenswürdigen[2] Instanz bestätigen zu lassen. Diese Instanz heißt »Zertifizierungsstelle« (*certificate authority* oder kurz »CA«), und die Bestätigung des Zusammenhangs nennt man »Zertifikat«. X.509 ist ein Standard dafür, wie solche Zertifikate aussehen. Man spricht in diesen Zusammenhang auch von einer *public-key infrastructure* oder PKI.

Zertifizierungsstelle
Zertifikat
PKI

Ein Zertifikat besteht im Wesentlichen aus vier Komponenten:

- Einem Namen für die Person (oder den Server), von dem die Rede ist. X.509 verwendet dafür sogenannte *distinguished names* der Form

```
cn=Hugo Schulz,o=Beispiel GmbH,l=Musterdorf,c=DE          Person
cn=www.example.com,o=Beispiel GmbH,l=Musterdorf,c=DE      Server
```

- Einem öffentlichen Schlüssel, der zu der benannten Person (oder dem benannten Server) gehört. Genau diesen Zusammenhang soll das Zertifikat bestätigen.
- Eine digitale Signatur für das Zertifikat. Diese sichert dessen Authentizität.
- Einen Namen (DN) für die Zertifizierungsstelle.

Sie können die Authentizität eines solchen Zertifikats prüfen, indem Sie sich den öffentlichen Schlüssel der Zertifizierungsstelle besorgen (der dazugehörige Name steht ja im Zertifikat) und damit die Signatur nachrechnen. Diesen öffentlichen Schlüssel bekommen Sie natürlich auch in Form eines X.509-Zertifikats.

Wenn Sie clever sind, dann fragen Sie sich an dieser Stelle, wer wohl das Zertifikat der Zertifizierungsstelle signiert haben mag, um zu dokumentieren, dass *dieses* echt ist. Die Antwort darauf lautet: Das macht die Zertifizierungsstelle selber. Bevor Ihnen das komisch vorkommt (was im ersten Moment absolut entschuldbar wäre), sollten Sie sich überlegen, dass Sie in diesem Geschäft *irgendwem* vertrauen müssen. Da Sie der Zertfizierungsstelle glauben, dass sie zum Beispiel Alices Zertifikat zuverlässig signieren

[2] »Vertrauenswürdig« heißt in einem Kontext wie diesem gemäß Peter Gutmann dasselbe wie »man verläßt sich drauf, weil man ohnehin keine andere Wahl hat.«

kann, können Sie ihr auch glauben, dass sie ihr eigenes Zertifikat zuverlässig signieren kann – das ist kein großer Schritt. Man spricht bei einem solchen von der Zertifizierungsstelle selbst signierten Zertifikat auch von einem »Wurzelzertifikat« (*root certificate*). Wurzelzertifikat

Wobei natürlich die Frage bleibt, woher Sie wissen, dass das Wurzelzertifikat *wirklich* echt ist und Ihnen nicht – komplett mit gültiger Signatur – von einem geschickten Angreifer untergeschoben wurde. Sie können dieser Sache natürlich nachgehen, indem Sie sich bei der Zertifizierungsstelle vergewissern – oder (was wahrscheinlicher ist) Sie vertrauen blind dem Hersteller Ihres Browsers, der Ihnen hilfreicherweise (?) ein paar Dutzend Wurzelzertifikate von verschiedenen Zertifizierungsstellen fest eingebaut und (hoffentlich …) vorher seine Hausaufgaben ordentlich gemacht hat.

Grundsätzlich gibt es auch die Möglichkeit, eine Hierarchie von Zertifizierungsstellen zu bilden. Das heißt, ein Zertifikat für eine Person oder einen Server kann mit einem Zertifikat signiert sein, das nicht direkt das Wurzelzertifikat einer Zertifizierungsstelle ist, sondern wiederum mit einem anderen Zertifikat signiert wurde. Diese »Kette« von Zertifikaten läßt sich weiterverfolgen, bis irgendwann ein selbstsigniertes Zertifikat erreicht ist. Hierarchie

Um einen Server über X.509 authentisieren zu können, brauchen Sie zumindest ein Zertifikat für diesen Server. Wenn es sich dabei um einen Web-Server handelt, werden Sie kaum anders können, als sich dieses Zertifikat von einer kommerziellen Zertifizierungsstelle wie VeriSign ausstellen zu lassen, damit die gängigen Browser es ohne weitere Mühe verifizieren können. Server-Zertifikat

CAcert (`http://www.cacert.org/`) ist eine Organisation, die auf nichtkommerzieller Basis (unter anderem) Zertifikate für Server ausstellt. CAcert arbeitet im Moment an einer Akkreditierung durch die wesentlichen Browser-Hersteller; dieser Prozess ist aktuell (Mai 2011) aber noch nicht abgeschlossen. CAcert

Wenn Sie Ihre Zertifikate nur für »interne« Zwecke brauchen – etwa für eine VPN-Infrastruktur oder einen Web-Server im Intranet –, gibt es keinen vernünftigen Grund, VeriSign und Konsorten Geld in den Rachen zu werfen. Sie können ohne weiteres selbst als Zertifizierungsstelle für Ihre eigenen Zertifikate agieren und so nicht nur jede Menge Kosten sparen, sondern auch eine wesentlich sicherere Infrastruktur aufbauen (da Sie sich selbst deutlich mehr vertrauen dürften als den Windhunden vom kommerziellen Zertifizierungsmarkt). Eigene Zertifizierungsstelle

Im Rest dieses Kapitels erklären wir, wie Sie X.509-Zertifikate auf der Basis von OpenSSL (`http://www.openssl.org/`) verwalten können. Es gibt alternative frei verfügbare Implementierungen von X.509 sowie bequemere Oberflächen zur Verwaltung von Zertifizierungsstellen, aber es handelt sich auch nicht um Hexenwerk.

B.2 Eine Zertifizierungsstelle generieren

Um zur Zertifizierungsstelle zu werden, müssen Sie ein Schlüsselpaar (privater und öffentlicher Schlüssel) für die Zertifizierungsstelle erzeugen und damit ein selbstsigniertes Wurzelzertifikat generieren. Im Idealfall tun Sie das aus Sicherheitsgründen auf einem Rechner, den Sie für nichts Anderes brauchen – ein Laptop-Computer, den Sie sicher wegschließen, wenn Sie nicht gerade ein Zertifikat ausstellen müssen, ist am besten. Schlüsselpaar Sicherheit

Da alles Nötige auf der Kommandozeile stattfinden kann, reicht als Rechner für die Zertifizierungsstelle irgendeine alte Möhre, die sonst keiner mehr benutzen möchte, dicke aus. Machen Sie für den Fall des Falles Sicherheitskopien, die Sie mindestens genauso sicher aufbewahren wie den Rechner.

Manche Linux-Distributionen – etwa der »SUSE Linux Enterprise Server« von Novell – bieten hilfreicherweise an, im Rahmen der Installation Ihres Web-Servers, LDAP-Servers, … auch gleich eine Zertifizierungsstelle mitzugenerieren. Sowas lehnen Sie natürlich dankend ab.

Verwaltungsinformationen

Außerdem müssen Sie als Zertifizierungsstelle einige Verwaltungsinformationen speichern, etwa eine Liste der Zertifikate, die Sie ausgestellt haben. Im einfachsten Fall legen Sie für die Zertifizierungsstelle einen eigenen Benutzer an, in dessen Heimatverzeichnis Sie die benötigte Dateistruktur etablieren:

```
# useradd -m ca
# /bin/su - ca
$ mkdir private certs
$ chmod 700 private certs
$ echo 01 >serial
$ touch index.txt
```

Die Datei `serial` enthält die »Seriennummer« des nächsten zu erzeugenden Zertifikats und in `index.txt` steht die Zertifikatsliste. Im Verzeichnis `private` legen wir den privaten Schlüssel der Zertifizierungsstelle ab, und in `certs` landen die von der Zertifizierungsstelle ausgestellten Zertifikate.

Für die spätere Arbeit ist es am günstigsten, eine Konfigurationsdatei anzulegen, in der die wichtigsten Parameter für die verschiedenen OpenSSL-Werkzeuge stehen. Dies vermeidet Verwirrung und Fehler durch Vergesslichkeit und macht die Vorgänge nachvollziehbar. Eine mögliche Konfigurationsdatei sehen Sie in Bild B.1.

Wurzelzertifikat erzeugen

Wenn Sie die Konfigurationsdatei aus Bild B.1 in `/home/ca/openssl-ca.cnf` abgelegt haben, können Sie das Wurzelzertifikat für die Zertifizierungsstelle wie folgt erzeugen:

```
$ export OPENSSL_CONF=~/openssl-ca.cnf
$ openssl req -x509 -days 1825 -newkey rsa:2048 -out ca-cert.pem
Generating a 2048 bit RSA private key
.............................................+++
.....................+++
writing new private key to '/home/ca/private/ca-key.pem'
Enter PEM pass phrase:geheim
Verifying - Enter PEM pass phrase:geheim
-----
$ _
```

Die Passphrase, die Sie hier festlegen, müssen Sie jedesmal wieder eingeben, wenn Sie mit dieser Zertifizierungsstelle ein neues Zertifikat für einen Benutzer oder Server ausstellen wollen.

Es versteht sich von selbst, dass Sie im wirklichen Leben eine deutlich komplexere Passphrase verwenden wollen als in unserem Beispiel. Von der Sicherheit des privaten Schlüssels Ihres Wurzelzertifikats hängt die Sicherheit Ihrer kompletten Zertifikatsinfrastruktur ab!

Mit der `-days`-Option geben Sie an, wie lang das Wurzelzertifikat gilt. Da mit dem Ablauf dieses Zertifiakts die von Ihrer Zertifizierungsstelle ausgestellten Zertifikate nicht mehr verifiziert werden können, sollten Sie diese Dauer mit Bedacht wählen. Eine zu kurze Gültigkeitsdauer zwingt Sie dazu, schon bald alle ausgestellten Zertifikate zu erneuern, während eine zu lange Gültigkeitsdauer im Extremfall bedeuten kann, dass Fortschritte in der Computertechnik es möglich machen, den öffentlichen Schlüssel der Zertifizierungsstelle zu brechen, und die Zertifizierungsstelle dadurch kompromittiert wird. Mit einem 2048-Bit-Schlüssel sollten Sie allerdings für die vorhersehbare Zukunft Ruhe haben.

```
[ca]
default_ca = CA

[CA]
dir              = /home/ca
certificate      = $dir/ca-cert.pem                    CA-Zertifikat
private_key      = $dir/private/ca-key.pem             Privater Schlüssel
database         = $dir/index.txt                      Zertifikatsliste
new_certs_dir    = $dir/certs                          Neue Zertifikate
serial           = $dir/serial                         Seriennummer

default_crl_days = 7                                   Rhythmus für CRL-Veröffentlichung
default_days     = 730                                 Gültigkeitsdauer für Zertifikate
default_md       = sha1                                Prüfsumme für Zertifikate

policy           = CA_policy
x509_extensions  = certificate_extensions

[CA_policy]                                            Namensregeln für die Zertifikate
commonName             = supplied                      Muss angegeben sein
emailAddress           = supplied
organizationalUnitName = optional                      Darf fehlen
organizationName       = match                         Muss mit CA-DN übereinstimmen
localityName           = match
countryName            = match

[certificate_extensions]
basicConstraints = CA:false

[req]                                                  Regeln für CA-Zertifikat
default_bits       = 2048
default_keyfile    = /home/ca/private/ca-key.pem
default_md         = sha1
prompt             = no
distinguished_name = CA_dn
x509_extensions    = CA_extensions

[CA_dn]                                                DN für die Zertifizierungsstelle
commonName       = CA
emailAddress     = ca@example.com
organizationName = Beispiel GmbH
localityName     = Musterhausen
countryName      = DE

[CA_extensions]
basicConstraints = CA:true
```

Bild B.1: Konfigurationsdatei für eine OpenSSL-basierte CA

Sie können das gerade erstellte Zertifikat mit dem Kommando

```
$ openssl x509 -in ca-cert.pem -text -noout
```

anschauen.

B.3 Server-Zertifikate generieren

Der Prozess für die Erzeugung von Server-Zertifikaten ist derselbe, egal ob Sie ein Zertifikat von einer kommerziellen Zertifizierungsstelle wünschen oder ob Sie das Zertifikat selbst ausstellen:

CSR

1. Sie (als Inhaber des Servers) erzeugen ein Schlüsselpaar und auf dieser Basis einen *Certificate Signing Request* (CSR) – eine Datei, die Ihren öffentlichen Schlüssel und einen DN für den Server enthält. (Den privaten Schlüssel aus dem Schlüsselpaar behalten Sie natürlich für sich.)
2. Die Zertifizierungsstelle überzeugt sich davon, dass Ihr CSR vernünftig aussieht, stellt das dazugehörige (signierte) Zertifikat aus und schickt es Ihnen zurück.
3. Sie installieren das Zertifikat auf Ihrem Server.

Wenn Sie Ihre eigene Zertifizierungsstelle sind, dann übernehmen Sie den zweiten Schritt natürlich selbst.

Schlüsselpaar und CSR erzeugen Wir müssen Ihnen also als erstes erklären, wie Sie ein Schlüsselpaar und einen CSR generieren. Das funktioniert wieder mit OpenSSL:

```
$ unset OPENSSL_CONF
$ cd /tmp
$ openssl req -newkey rsa:2048 -keyout server-key.pem -out server-csr.pem
Generating a 2048 bit RSA private key
..........................+++
........+++
writing new private key to 'server-key.pem'
Enter PEM pass phrase:abc123
Verifying - Enter PEM pass phrase:abc123
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:Musterhausen
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Beispiel GmbH
Organizational Unit Name (eg, section) []:↩
Common Name (eg, YOUR name) []:www.example.com
Email Address []:webmaster@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:↩
```

```
An optional company name []:↩
$ _
```

Was Sie als Bestandteile des DN eingeben, ist wahlfrei, solange es zur »Policy« der Zertifizierungsstelle passt (siehe Bild B.1).

In unserem Beispiel kann der »state or province name« leer bleiben, weil die Policy darüber keine Aussagen macht. Ebenso darf der »organizational unit name« leer bleiben (oder einen beliebigen Wert haben). Der »organization name« muss mit dem im Wurzelzertifikat der Zertifizierungsstelle übereinstimmen. Der »common name« ist grundsätzlich beliebig, aber *muss* für ein Server-Zertifikat dem FQDN des betreffenden Servers entsprechen.

Beim Erzeugen des Schlüsselpaars besteht OpenSSL darauf, dass Sie eine Passphrase zur Verschlüsselung des privaten Schlüssels eingeben. Grundsätzlich ist das lobenswert, aber es gibt Situationen – etwa wenn der Schlüssel für einen Server gebraucht wird, der auch starten soll, ohne dass jemand an der Konsole steht und die Passphrase eingibt –, wo Sie vielleicht lieber einen privaten Schlüssel *ohne* Passphrase hätten. Für diesen Fall können Sie wie folgt eine unverschlüsselte Kopie Ihres privaten Schlüssels erzeugen:

```
$ openssl rsa -in server-key.pem -out server-key-np.pem
Enter pass phrase for server-key.pem:abc123
writing RSA key
```

Anschließend steht der unverschlüsselte private Schlüssel in `server-key-np.pem`. Behandeln Sie ihn mit Sorgfalt.

Wenn Sie direkt einen nicht verschlüsselten privaten Schlüssel erzeugen wollen, können Sie »`openssl req`« mit der Option `-nodes` aufrufen.

Zertifikat besorgen Den CSR schicken Sie entweder an eine Zertifizierungsstelle Ihres Vertrauens oder Sie stellen sich selbst ein Zertifikat aus. Für Letzteres müssen Sie im Heimatverzeichnis des Benutzers `ca` stehen, und die Umgebungsvariable `OPENSSL_CONF` muss auf die Konfigurationsdatei für die Zertifizierungsstelle zeigen:

```
$ cd                                                    Zurück ins Heimatverzeichnis
$ export OPENSSL_CONF=$HOME/openssl-ca.cnf
$ openssl ca -in /tmp/server-csr.pem
Using configuration from /home/ca/openssl-ca.cnf
Enter pass phrase for /home/ca/private/ca-key.pem:geheim
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
localityName          :PRINTABLE:'Musterhausen'
organizationName      :PRINTABLE:'Beispiel GmbH'
commonName            :PRINTABLE:'www.example.com'
emailAddress          :IA5STRING:'webmaster@example.com'
Certificate is to be certified until May  3 10:00:56 2013 GMT (730 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Certificate:
  ◁◁◁◁◁
Data Base Updated
$ _
```

Das neue Zertifikat landet dann im Verzeichnis `certs`. Die Zertifikate dort sind nach ihrer Seriennummer benannt (das gerade angelegte Zertifikat heißt `certs/01.pem`), so dass Sie gegebenenfalls in der Datei `serial` nachschauen müssen, was die letzte vergebene Nummer war. (Denken Sie daran, dass `serial` immer die Nummer des *nächsten* Zertifikats enthält. Wenn Sie schlau sind, schauen Sie statt dessen in die Datei `serial.old`.) (Denken Sie auch daran, dass die Seriennummern hexadezimal sind.)

Das neue Zertifikat (und möglicherweise den dazugehörigen privaten Schlüssel) sollten Sie anschließend auf den gewünschten Rechner kopieren. Wenn Sie – wie empfohlen – einen dedizierten Rechner für die Zertifizierungsstelle verwenden, dann involviert dieser Prozess sinnvollerweise einen USB-Stick, denn es ist besser, wenn der Rechner mit der Zertifizierungsstelle nicht ans Netz angeschlossen ist.

Client-Zertifikate

Client-Zertifikate Zertifikate für Benutzer – sogenannte Client-Zertifikate – können Sie übrigens ganz analog erstellen. Dafür verwenden Sie einfach einen CSR, bei dem das »common name«-Feld keinen FQDN enthält, sondern den Namen der betreffenden Person.

Im Idealfall erzeugen nicht Sie diesen CSR (und das dazugehörige Schlüsselpaar), sondern der spätere Inhaber des Zertifikats. Nur auf diese Weise kann sicher gestellt werden, dass der private Schlüssel des Benutzers wirklich privat bleibt.

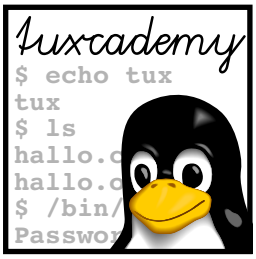
PKCS#12

Web-Browser erwarten Client-Zertifikate gerne im »PKCS#12«-Format, das das Zertifikat und den dazugehörigen privaten Schlüssel zusammenfasst. Wenn Sie sowohl das Zertifikat als auch den privaten Schlüssel haben, können Sie ein PKCS#12-Zertifikat mit dem Kommando

```
$ openssl pkcs12 -export -in hugo-cert.pem -inkey hugo-key.pem -out hugo.p12
Enter pass phrase for hugo-key.pem:x8o,AqS!k
Enter Export Password:foo-bar
Verifying - Enter Export Password:foo-bar
```

erzeugen.

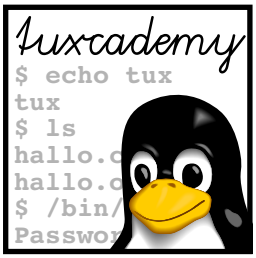
Das »Export Password« müssen Sie wieder angeben, wenn Sie das PKCS#12-Zertifikat mit dem Browser einlesen. Es muss (bzw. sollte) nichts mit dem Kennwort für den privaten Schlüssel zu tun haben.



C

Kommando-Index

Dieser Anhang fasst alle im Text erklärten Kommandos zusammen und verweist auf deren Dokumentation sowie die Stellen im Text, wo die Kommandos eingeführt werden.



Index

Dieser Index verweist auf die wichtigsten Stichwörter in der Schulungsunterlage. Besonders wichtige Stellen für die einzelnen Stichwörter sind durch **fette** Seitenzahlen gekennzeichnet. Sortiert wird nur nach den Buchstaben im Indexeintrag; „`~/.bashrc`“ wird also unter „B“ eingeordnet.