SelfLinux-0.12.3

Grundlagen Sicherheit

Autor: Gabriel Welsche (*gabriel.welsche@web.de*)
Autor: Karsten Schulz (*kaschu@t800.ping.de*)
Formatierung: Matthias Hagedorn (*matthias.hagedorn@selflinux.org*)

Inhaltsverzeichnis

1 Einleitung und Überblick

1.1 Allgemeines

Dieses Kapitel soll einen groben aber möglichst umfassenden Überblick zur Systemsicherheit des Linux-Rechners vermitteln. Das Themengebiet Sicherheit ist sehr komplex und besitzt zahlreiche Facetten. In der Regel reicht es nicht aus, irgendwelche **Firewall-Konfigurationsskripte** zu kopieren und auf einem Linux-Rechner zu starten, um diesen **sicher** zu machen.

Denn bevor man jedoch überstürzt loslegt, sollte man sich über verschiedene Dinge klar werden:

* Was verstehe ich unter Sicherheit?
* Was kann ich tun, um meine Sicherheit zu erreichen?
* Wie kann ich meinen Erfolg messen?
* Wie kann ich meine Sicherheit aufrechterhalten?

1.2 Was ist Sicherheit?

Der Begriff Sicherheit beschreibt keinen konkreten Zustand eines Systems. Sicherheit steht dafür, dass nichts und niemand unerwünschte Aktionen durchführt oder gewünschte Aktionen untersagt. Sicherheit herrscht also dann, wenn **alles so bleibt, wie es sein soll**.

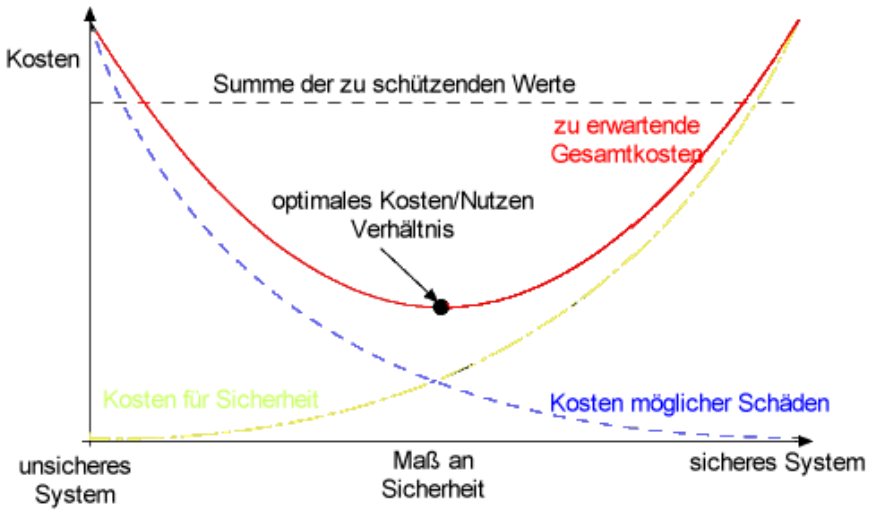
Man muss für seine eigene Sicherheit demnach nur noch definieren, was **alles**, **so bleibt** und **sein soll** bedeuten!

Gehen wir der Reihe nach vor: was bedeutet **alles**? Vermutlich gehört dazu der Linux-Rechner, um den es hier geht. Außerdem gehören die Daten auf dem Rechner dazu. Alle Benutzer des Rechners, aber auch das LAN, in dem sich der Rechner vielleicht befindet, gehören dazu.

Leider ist das Wörtchen **alles** damit noch nicht erschöpfend untersucht. Denn es gehört auch die Haustür, die nur diejenigen einlässt, die in die Nähe Ihres Rechners dürfen, dazu.
Dazu gehört auch der Stromlieferant, der Ihren Rechner zuverlässig mit Energie versorgen soll. Und ganz besonders gehören auch Sie selbst dazu, der oder die den Linux-Rechner vernünftig konfigurieren will oder soll!

Diese Überlegungen kann man noch beliebig fortsetzen, um festzustellen, was denn nun zu **alles** gehört.

Spätestens jetzt wird hoffentlich deutlich, dass noch ein anderer Faktor eine Rolle bei der Sicherheit spielt: der Aufwand. Oder besser gesagt: das Verhältnis zwischen dem Aufwand, der betrieben werden muss, und dem Nutzen, der damit erreicht werden soll.



Kosten und Nutzen

Bei der Beschäftigung mit dem Thema Sicherheit spielen also auch Kosten und Nutzen eine Rolle.

Der erste Schritt, um sein System sicher zu machen, ist also, den Zustand zu definieren, den das System einhalten soll. Womit wir beim nächsten Aspekt der Überlegungen wären. Die Definition, wie das System sein soll, könnte beispielsweise folgende Punkte enthalten:

* Es soll 24 Stunden am Tag, 7 Tage in der Woche arbeiten.
* Es können sich nur legitime Systembenutzer anmelden.
* Es werden keine externen Dateien vom Systembenutzer eingebracht.
* Systembenutzer haben starke Passwörter.
* Das System ist frei von Viren, Trojanern oder sonstigem Getier.
* Der Datenzugriff erfolgt entsprechend der Berechtigung des Systembenutzers.
* Es gibt keine Möglichkeit, sensible Daten zu stehlen.
* Netzwerkzugriffe außerhalb der Geschäftszeiten werden protokolliert.
* ...

Diese Auflistung sollte so umfangreich gehalten werden, dass sämtliche Aspekte des Systembetriebes durch solche Vorgaben beschrieben worden sind. Diese Vorgaben heißen auch Sicherheitsrichtlinien beziehungsweise *security policy*.

Aus diesen Sicherheitsrichtlinien ergeben sich die Konfigurationen und die Maßnahmen, um diesen Soll-Zustand zu verwirklichen. Zu den oben angeführten Punkten passen demnach folgende Aktionen:

* Rechner an einer unterbrechungsfreien Stromversorgung anschließen und ein Backup-System im Standby-Modus betreiben.
* korrekte Pflege der Benutzerdatenbank durch den Administrator
* physische Sicherung z.B. der internen Laufwerke (Schloss oder Ausbau), Absicherung der Schnittstellen (z.B. USB), Sicherung der Netzwerkdosen, am besten alle Server eines Bereiches in einem eigenen Raum
* organisatorische Sicherung (Putzkräfte stehen unter Kontrolle und arbeiten in sensiblen Bereichen wie Serverräumen nur tagsüber, am besten unter Aufsicht)
* Schulung der Systembenutzer, Überprüfen der Kennworte durch Passwort-Crackprogramm

* ständige Aktualisierung der Virenscanner
* Schutz vor Zugriff von Außen
* Anpassung der Topologie (Netzinfrastruktur)
* ...

Erst jetzt können Sie feststellen, ob Ihr System sicher im Sinne Ihrer Vorgaben ist. Erst jetzt, wo Sie wissen, *wie* es sein soll, können Sie eine konkrete Aussage treffen. Entweder "ja, das System befindet sich in dem Zustand, in dem es sein soll" oder "Nein, ein Parameter ist nicht so, wie er sein soll, es besteht Handlungsbedarf!"

Als abschließende Betrachtung fehlen nun nur noch die Maßnahmen, mit denen Sie dafür sorgen, dass alles auch so **bleibt**, wie Sie es eingerichtet haben. Dazu müssen Sicherungen und Protokollsysteme eingerichtet und aktiviert werden. Und merken Sie sich:

> **Sicherheit ist kein Produkt, sondern ein Prozess!**

1.3 Kapitelübersicht

Dieses Grundlagenkapitel soll einen Einblick in das Thema IT-Sicherheit geben. Nach einer kurzen Einleitung wird im Abschnitt ▶ Sicherheitsprozess eine systematische Vorgehensweise zur Durchsetzung und Aufrechterhaltung eines gewünschten Sicherheitsniveaus vorgestellt.

Im darauf folgenden Unterkapitel soll ein Überblick über ▶ Bedrohungen vermittelt werden. Sicherlich kennen Sie ▶ Viren und haben auch schon einmal etwas über ▶ Trojanische Pferde gehört, aber wissen Sie auch, was ein ▶ Sniffer oder was ▶ Spoofing ist und wie man sich davor schützen kann? Wir werden in diesem Kapitelabschnitt auch ▶ Buffer Overflows (Pufferüberläufe), ▶ Netzwerk-Scan-Techniken, ▶ DOS- und ▶ DDOS-Attacken ansprechen.

Im Unterkapitel ▶ Allgemeine Schutzmaßnahmen werden die Grundvoraussetzungen für die **Härtung** des Linux-Systems gegenüber potentiellen Angriffen geschaffen. Dazu gehören beispielsweise die Deinstallation nicht benötigter Pakete, die Absicherung des Boot-Vorgangs und der Schutz der Systemlogfiles.

Fortgesetzt wird dieses Kapitel mit einem Abschnitt über ▶ Authentisierung, Autorisierung und Zugriffssteuerung mit PAM. PAM ist eine englische Abkürzung für "**P**lugable **A**uthentication **M**odules" heißt auf deutsch: steckbare Authentifikationsmodule.

Anschließend folgt ein Unterkapitel zum ▶ Superserver xinetd, indem auch die ältere Variante `xinetd` + TCP_Wrapper `tcpd` besprochen wird. Superserver steuern und überwachen Netzwerkdienste wie z.B. FTP, Login oder Samba. Trifft eine Anfrage auf einem vom Superserver verwalteten Port ein, so wird der entsprechende Serverprozess (z.B. der FTP-Server `ftpd`) gestartet. Zuvor wird jedoch nach bestimmten Regeln entschieden, ob der Netzwerkzugriff erlaubt ist oder nicht. Diese Zugriffssteuerung ist das Thema dieses Abschnittes.

Ein weiteres Unterkapitel beschäftigt sich mit ▶ Firewalls. Darin werden Paketfilter aber auch Kernelparameter und Proxies eine Rolle spielen.

Nachdem in den vorangegangenen Abschnitten Methoden zur Verbesserung der System-Sicherheit behandelt wurden, ist das Thema eines weiteren Unterkapitels das ▶ Testen der Sicherheitsmaßnahmen.

Anschließend folgen zwei Unterkapitel, die sich mit dem Aufrechterhalten der Sicherheit beschäftigen. Wir beginnen mit dem ▶ Logging und Accounting, also der Nachvollziehbarkeit dessen, was auf dem System passiert. Der für Systemmeldungen zuständige Syslog-Dienst wurde in einem ausgegliederten Kapitel betrachtet, deshalb wollen wir uns hier auf die Verwaltung der Log-Dateien und deren Schutz vor Angriffen beschränken.

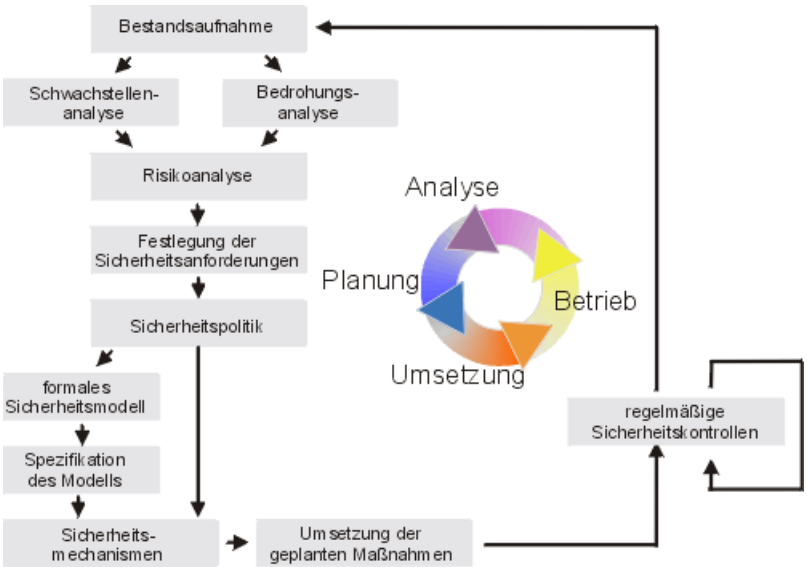
Ein weiterer Abschnitt widmet sich dem Thema Einbruchserkennung (Intrusion Detection), einem sehr jungen Teilgebiet der IT-Sicherheit, welches Methoden zur frühzeitigen Erkennung von Angriffen bereitstellt. Man kann dies auch mit einer Alarmanlage vergleichen.

Sollte tatsächlich ein Einbruch festgestellt werden, hilft der Abschnitt Notfallplan im Falle einer Systemkompromittierung weiter. Um welche Maßnahmen es sich dabei im Einzelnen handeln könnte, soll in diesem vorletzten Unterkapitel behandelt werden.

Am Ende steht ein kurzes Fazit und eine Aufzählung der Themenbereiche, die zwar wichtig sind aber trotzdem (noch) keinen Eingang in dieses Kapitel gefunden haben.

2 Sicherheitsprozess

Um ein gewünschtes Sicherheitsniveau durchzusetzen und aufrechtzuerhalten, bedarf es einem geplanten und organisierten Vorgehen aller Beteiligten. Diese systematische Vorgehensweise wird in einem Prozessmodell beschrieben. Natürlich gibt es einige Sicherheitsprozesse, die jedoch meist an dem des BSI (Bundesamt für Sicherheit in der Informationstechnik) angelehnt sind. Wir wollen den folgenden Prozess vorstellen:



Sicherheitsprozess

Begonnen wird mit einer Bestandsaufnahme gefolgt von der Schwachstellenanalyse. Daraus ergeben sich mögliche Gefahren für das System, welche in einer Bedrohungsanalyse adressiert werden. Wenn die Bedrohungen des Systems bekannt sind, wird für jede Bedrohung das Risiko bestimmt und nach der zu erwartenden Schadenshöhe priorisiert. Aus dieser Risikoanalyse leiten sich die notwendigen Sicherheitsanforderungen ab, welche auch formal in einem Sicherheitsmodell spezifiziert werden können. Die Anforderungen werden durch Sicherheitsmechanismen wie

* Zugriffssteuerung
* Kryptographische Mechanismen
* Authentisierungsverfahren

umgesetzt. Letztendlich muss dieses damit erreichte Sicherheitsniveau gehalten werden. Dies passiert durch regelmäßige (automatisierte) Sicherheitskontrollen und das zyklische Durchlaufen dieses Sicherheitsprozesses.

Da die genaue Beschreibung eines Sicherheitsprozesses dieses Kapitel sprengen würde, haben wir diese theoretischen Grundlagen in ein eigenständiges Kapitel ausgelagert, welches im nächsten Release erscheinen wird.

3 Bedrohungen und Schwachstellen

3.1 Einführung Bedrohungen und Schwachstellen

Zu Beginn jedes Sicherheitsprozesses steht die Schwachstellen- und Bedrohungsanalyse. Dies erfordert allerdings genaues Wissen über potentielle Gefahren. Deshalb möchten wir in diesem Kapitel eine Übersicht vermitteln.

3.2 Systemanomalien

Um den Begriff der Systemanomalie beschreiben zu können, muss man sich über zwei Verhaltensweisen klar werden:

1. *normales Systemverhalten:*
   Alle Systemkomponenten (Hardware + Software) erfüllen die an sie gestellten Erwartungen unter bestimmten Rahmenbedingungen.
2. *anormales Systemverhalten:*
   Die Abweichung vom normalen Systemverhalten bezeichnet man als anormales Systemverhalten

Schwierig gestaltet sich die Einordnung in eine dieser Kategorien, viele Nutzer können aufgrund ihrer geringen Kenntnisse zu Fehlentscheidungen gelangen. Anormales Systemverhalten ist das sichtbare Resultat von Anomalien. Anders ausgedrückt: Als Anomalien bezeichnet man Veränderungen an Hard- und/oder Software bzw. deren Konfiguration, die durchaus schwere Schäden nach sich ziehen können. Man unterscheidet Systemanomalien erster, zweiter und dritter Art:

Systemanomalien der ersten Art

Anormales Verhalten kann durch eine Vielzahl an Faktoren hervorgerufen werden. Wenn dieses Verhalten weder beabsichtigt noch spezifiziert oder gar durch gewollte Systemveränderungen hervorgerufen wurde, handelt es sich um Systemanomalien erster Art. Dazu zählen insbesondere Umwelteinflüsse wie Überspannung durch Blitzschlag oder Übertragungsfehler. Zu den Anomalien der ersten Art gehören aber auch menschliche Fehler, die zum Beispiel bei der Programmierung auftreten und durch Angreifer ausgenutzt werden können (z.B. Pufferüberläufe).

Systemanomalien der zweiten Art

Bei diesen Anomalien wurden Komponenten (Betriebssystemmodule, Softwareklassen, Bibliotheken, Hardwarekomponenten, ...) absichtlich durch zusätzliche, schädliche Funktionen erweitert. Bekannte Vertreter sind Trojanische Pferde, die das Systemverhalten nach den Wünschen des Angreifers manipulieren.

Systemanomalien der dritten Art

Aus Anomalien zweiter Art entstehen durch die Erweiterung um Funktionen zur Reproduktion Systemanomalien dritter Art. Dazu gehören Bakterien (Ausbreitung begrenzt auf lokales System), Viren und Computerwürmer.

3.3 Viren

Ein Computervirus ist ein Stück Selbstreplizierende Software, die sich jedoch immer in ein ausführbares

Wirtsprogramm oder eine Systemkomponente hineinmogelt und auf einem beschreibbaren Medium abgespeichert ist.

Viren stellen für Linux (noch) keine Bedrohung dar. Trotzdem gibt es bereits welche, und nicht nur das, man findet sogar eine Anleitung zum Schreiben von Viren. Bekannteste Vertreter der Linux-Viren sind: Bliss, Staog, Telf, SIILOV.

Gegenmaßnahmen sind einerseits Virenscanner, z.B.

* fprot: http://www.fprot.org/
* AvGate: http://antivir.de
* AMaViS Mail Scanner: http://www.amavis.org/
* InterScan VirusWall von Trendmicro: http://de.trendmicro-europe.com/

und andererseits der Vergleich der Signaturen beim Installieren von Software, welche natürlich ausschließlich von vertrauenswürdigen Stellen stammen darf. Besonders vorsichtig sollte man als Superuser root agieren, aber das versteht sich ja von selbst.

3.4 Würmer

Würmer sind im Gegensatz zu Viren eigenständige Programme, die sich ebenfalls selbst replizieren können. Schon vor mehr als drei Jahrzehnten traten einzelne Exemplare auf Großrechnern auf. Diese wurden liebevoll "Kaninchen" genannt, weil sie sich so schnell vermehrten. Sie reproduzierten sich im Hauptspeicher und stahlen somit anderen Nutzern wertvolle Ressourcen. Sie wurden aber auch für nützliche Zwecke wie dem Einsammeln von Informationen eingesetzt, heute würde man solche Programme dann doch lieber als Vorfahren Mobiler Agenten ansehen.

Einer der bekanntesten Schädlinge war der Morris Wurm (1988), welcher zwischen 2000 und 6000 Internetrechner lahm legte. Das war ein sehr großer Teil des damaligen Internet. Anfällig waren neben BSD-Unix-Systemen auch alle DEC-Rechner und SUN3-Systeme. Computer mit "UNIX System V" als Betriebssystem waren "nur" anfällig, wenn mit Sendmail, `fingerd` und `rexec` eine Kompatibilität zu BSD bestand. Der Entwickler des Wurmes, Robert Tappan Morris Jr, wurde zu 10.000 Dollar, 3 Jahren Haft und 400 Stunden gemeinnütziger Arbeit verurteilt. Zusätzlich musste er die 150.000 Dollar Gerichtskosten tragen.

Man sieht, dass Computerwürmer ein wesentlich größeres Gefahrenpotential als Viren darstellen, gerade auch im Hinblick auf breitbandige Internetzugänge im privaten Bereich. Einen gezielten Schutz gegen Würmer gibt es nicht, man sollte versuchen, sein System so sicher wie möglich zu gestalten und kontinuierlich Sicherheitsupdates durchzuführen.

3.5 Trojanische Pferde

Der Begriff des trojanischen Pferdes stammt aus der griechischen Sagenwelt (Odyssee des Homer). Nach der vergeblichen Belagerung der Stadt Troja boten die Griechen ein riesiges hölzernes Pferd als Friedensgeschenk an. Nachdem dieses Holzpferd in die Stadt geschafft worden war, sprangen aus dem Inneren des Pferdes dutzende Soldaten und griffen die Stadt aus dem Hinterhalt an. Nur so gelang es den Griechen die Stadt zu erobern.

Genauso hinterhältig sind trojanische Pferde im Computerbereich. In durchaus nützliche Programme (wie z.B. `lsmod`) werden schädliche Funktionen eingebaut und durch den getäuschten Benutzer selbst aktiviert. Dabei muss der Schaden nicht gleich im Zerstören von Daten liegen, manchmal wird auch einfach nur die Ausgabe wichtiger Informationen unterdrückt. Einige Trojanische Pferde nisten sich direkt im Kernel ein (als Modul) und könnten mit `lsmod` sichtbar gemacht werden. Um dies zu verhindern, manipulieren trojanische Pferde das

Programm `lsmod` derart, dass es den Eindringling nicht anzeigt.

Es gibt dutzende Varianten, die hier niemals alle behandelt werden können. Als Schutzmaßnahmen kommen vor allem Einbruchserkennungssysteme zum Einsatz, insbesondere **Hostbasierte Intrusion Detection Systeme**. Des weiteren sollte man ausschließlich vertrauenswürdige Software installieren und vorher die digitale Signatur (Fingerprint) vergleichen.

3.6 Sniffer

Als Sniffer bezeichnet man ein Programm (oder ein Gerät), welches den Netzwerkverkehr abhört, protokolliert und in einer menschenlesbaren Form ausgibt. Gefährlich sind Sniffer vor allem beim Austausch von unverschlüsselten Nachrichten wie Passwörtern, Kreditkartennummern und vertraulichen Dateien.

Sniffer nutzen den *Promiscuous Mode* einer Netzwerkschnittstelle und empfangen so alle auf dem Netz übertragenen Daten, egal für wen diese bestimmt sind. Weder Sender noch Empfänger merken etwas von dieser Spionage.

Sicherheitsmaßnahmen: Vertrauliche Daten wie Kennwörter oder Kreditkartennummern sollten stets über verschlüsselte Verbindungen übertragen werden (`https`, `ssh`). Durch sichere Netzwerktopologien (z.B. durch Trennung der Netzwerksegmente mittels Router, Switch, Bridge) und den Einsatz von Paketfiltern zur logischen Trennung kann man das Blickfeld eines potentiellen Sniffers stark einschränken. Um einen Sniffer ausfindig zu machen, fragt man an allen Rechnern des Netzwerkes den Status der Netzwerkkarte mit `ifconfig` ab. Das ist die einfachste Lösung. Es gibt aber auch AntiSniffer Werkzeuge wie beispielsweise SniffDet, oder kommerzielle Produkte wie AntiSniff.

3.7 Spoofing

Unter Spoofing versteht man das Erschleichen von Vertrauen unter Vorgabe einer falschen Identität. Im Computerbereich unterscheidet man:

* *IP-Spoofing* (Vorgabe einer falschen IP-Adresse)
  Gegenmaßnahme: Paketfilter
* *DNS-Spoofing* (Einstreuen einer falschen IP-Namenszuordnung ins DNS-Kommunikationssystem - Cache pollution)
  Gegenmaßnahme: digitale Unterschriften beim Austausch von DNS-Einträgen
* *ARP-Spoofing* (Einstreuen einer falschen IP-MAC-Adressenzuordnung im Ethernet; Überschreiben durch Push-Technik)
  Gegenmaßnahme: Ersetzen des ARP-Cache durch eine in einer Datei gespeicherten Liste
* *RIP-Spoofing* (Umleiten der Datenströme durch Einstreuen falscher Routing-Informationen)
  Gegenmaßnahme: Setzen des entsprechenden Kernelparameters
* *WWW-Spoofing* (URL Rewriting, man in the middle attack)
  Gegenmaßnahme: Kontrolle der Zertifikate verschlüsselter Verbindungen

Der bekannteste Fall ist wohl die *Kevin Mitnick Attacke*, die 1994 auf das Netz des Sicherheitsexperten Tsutomu Shimumura stattfand. *Mitnick* begann seinen Angriff mit mehreren Verbindungsanfragen an den `rlogin` Port des Servers seines Opfers, diese Anfragen besaßen allerdings gefälschte interne IP-Adressen. Aufgrund dieser Last brach der Server zusammen. Nachdem *Kevin Mitnick* die TCP-Sequenznummerngenerierung analysiert hatte, konnte er unter Vorgabe der Identität des zusammengebrochenen Servers eine TCP-Verbindung aufbauen und den Zugang für zukünftige Angriffe weit öffnen.

Kevin Mitnick wurde verhaftet und bekam 1999 eine Haftstrafe von 48 Monaten (er saß aber schon zu diesem Zeitpunkt mehrere Jahre in Untersuchungshaft). Im Januar 2000 wurde er mit der Auflage entlassen, dass er drei

Jahre lang (also bis Januar 2003) weder Computer noch Mobiltelefon oder ähnliche Gerätschaften benutzen durfte.

Spoofing ist ein wichtiger Bestandteil der meisten Einbruchsversuche. Durch die Angabe einer falschen Identität können fremde Verbindungen entführt (Hijacking) oder beendet werden, und das Nachvollziehen eines Einbruchs / Einbruchversuchs gestaltet sich äußerst schwierig. Deshalb wird dieses Thema zukünftig in einem gesonderten Kapitel behandelt werden.

3.8 Netzwerk-Scan-Techniken

3.8.1 Netzwerk-Scan-Techniken Allgemeines

Als Scanning bezeichnent man den Versuch, möglich viele oder gar alle Rechner eines Netzwerkes zu verifizieren. Dies beinhaltet vor allem auch die Bestimmung der darauf laufenden Dienste. Folgende Informationen sind Ziel eines solchen Angriffs:

* IP-Adressen von Rechnern (die ans Internet angeschlossen sind)
* TCP/UDP Portnummern
* Systemarchitektur (x86, PowerPC, Sparc)
* Betriebssystem (Kernelversion, Windowsversion)

Dem geht meist die Erstellung eines System- oder Netzwerkprofiles voraus, in dem allgemeine Informationen wie geographischer Standort, Namen, Telefonnummern, IP-Adressen (Bereiche) und Dienste wie DNS-Server, Mail-Server oder Web-Server festgehalten werden.

Security Scanner bieten feinere Techniken zum Ausspionieren sicherheitsrelevanter Informationen eines Systems. Zu den damit gewonnenen Informationen zählen zum Beispiel:

* Benutzer- und Gruppennamen
* Schwachstellen, welche ausgenutzt werden können
* Routing-Tabellen
* SNMP-Informationen

3.8.2 Online-Check

Das Ziel dieses Angriffes besteht darin, herauszufinden, welche Rechner des Opfernetzwerkes (eines bestimmten IP-Bereiches) gerade aktiv sind. Dies ist möglich durch

* ICMP-Echo-Anfragen (Ping, Broadcast)
* UDP /TCP Sweeps

Beides kann durch Kernelparameter bzw. Paketfilter verhindert werden.

3.8.3 Port-Scanning

Wenn nach einem erfolgreichen Online-Check die aktiven Rechner bekannt sind, wird der Angreifer meist die angebotenen Dienste mittels Port-Scanning ermitteln. Dazu bedient er sich oftmals der **TCP SYN Scan** Methode, bei der eine TCP-Verbindung nur angefragt aber nicht aufgebaut wird. Deshalb sind solche Attacken nur schwer zu finden und werden sehr selten in Logfiles protokolliert. Eine "Weiterentwicklung" sind die so genannten **Stealth Scans**, die auch Paketfilter passieren und trotzdem im Netzwerkverkehr unentdeckt bleiben. Sie missachten einfach das **Three Way Handshake** Protokoll und interpretieren die Antworten der

Opferrechner. Zu den Stealth Attacken gehören:

* XMAS Scan
* Null Scan
* Syn/Ack Scan
* Fin Scan

Port-Scans können durch Port-Scan-Detektoren erkannt werden, zu den wichtigsten zählen: Scanlogd PortSentry Natürlich gibt es auch wieder Möglichkeiten, diese Detektoren zu umgehen, darauf wollen wir an dieser Stelle nicht eingehen. Einen Hinweis zum Schluss: Theoretisch kann ein Angreifer einen Port-Scan-Detektor dazu benutzen, durch einen gefälschten Scan-Angriff (Vortäuschung eines Angriffs vom Gateway-Rechner --> Spoofing) den Opferrechner vom externen Netz zu trennen.

3.8.4 Nmap

Der Netzwerk-Scanner `nmap` unterstützt zahlreiche Angriffspraktiken zum Scannen von einzelnen Rechnern oder ganzen Netzwerken. Er bietet auch viele nützliche Funktionen für einen sinnvollen Einsatz. Denkbar wären beispielsweise die Überwachung der offenen Ports aller Netzwerkrechner oder das automatische Auslösen eines Alarms, wenn ein neuer Rechner ans Netz angeschlossen wird. Mit `nmap` und den zahlreichen Möglichkeiten wird sich ein eigenständiges Kapitel beschäftigen.

3.8.5 Betriebssystemerkennung

Systemschwachstellen sind meist an das verwendete Betriebssystem gebunden, deshalb sind Informationen darüber für den Angreifer äußerst wertvoll.

Viele Dienste wie z.B. `telnet`, `ssh` oder `www` geben auch nicht autorisierten Benutzern Betriebssysteminformationen. Sie können dies einfach nachvollziehen:

```
user@linux / $ telnet ip-opfer 80

Trying 217.72.195.42...
Connected to ha-42.web.de.
Escape character is '^]'.

user@linux / $ get

http/1.0 400 Bad Request
Date: Wed, 19 Jan 2000 15:33:21 GMT
Server: Apache/1.3.3 (Unix)   (Red Hat/Linux)
Connection: close
....
```

Aus der vorletzten Ausgabezeile ist ersichtlich, dass es sich um einen Linux Rechner mit Red Hat handelt. Es gibt mittlerweile einige Betriebssystemerkennungsprogramme wie `queso` oder `nmap`.

3.8.6 Firewalking

Darunter versteht man eine Technik, um Informationen von Netzwerken hinter einer Firewall zu erlangen. *David Goldsmith* und *Michael Schiffmann* haben diese Technik im Oktober 1998 veröffentlicht. Firewalking benutzt `traceroute`, das eigentlich entwickelt wurde, den Weg einer Verbindung aufzuzeigen. Duch die Angabe

eines Ausgangs- bzw Zielport können auch Firewalls passiert werden.

3.8.7 Inverse Mapping

Mit Inverse Mapping Techniken können Informationen über solche Rechner und Netzwerke erhalten werden, die als unerreichbar gelten (z.B. Rechner eines internen Netzwerkes).

Beispiel ftp-Bounce-Attacke:

Der Angreifer benutzt einen normalen ftp-Client und verbindet sich mit dem ftp-Server des Opfers. Er benutzt den ftp-Port-Befehl um die IP-Adresse und die Portnummer des anzugreifenden Dienstes anzugeben. Der Angreifer sendet nun in Kommandos, die der angegriffene Dienst versteht, in Form von Dateien an den ftp-Server und dieser leitet die Kommandos einfach an den verbundenen Dienst. Wenn der ftp-Server hinter der Firewall steht, sind durch diese Methode auch Scans des internen Netzwerkes möglich.

3.9 Buffer Overflows

Die wohl am häufigsten auftretende Schwachstelle sind Programme, die Pufferüberläufe (Buffer Overflows) ermöglichen. Diese Programme werden zu einer gefährlichen Bedrohung, wenn sie unter privilegierten Rechten ablaufen. Ein Angreifer kann durch das Ausnutzen dieser Schwachstelle beliebigen Programmcode zur Ausführung bringen, und zwar mit den selben Privilegien wie die des fehlerhaften Programms. Wenn also beispielsweise der Web-Server `apache` unter root Rechten ablaufen würde und eine solche Schwachstelle existiert, so kann ein Angreifer dem System den denkbar größtmöglichen Schaden zufügen.

Schutzmaßnahmen für Programmierer und Entwickler:

* Sichere Programmierung, Einsatz von "Source Code Security Analyzer" zur Feststellung möglicher Fehler
* Libsafe ersetzt unsichere C-Funktionen (z.B. `sprintf`, `strcp`, `gets`) durch Bibliotheksfunktionen
* Openwall-Patch: nichtausführbarer Stack Array-Grenzüberwachung

3.10 Denial of Service (DOS Attacken)

Angriffe mit dem Ziel, ein System oder einen Dienst lahm zu legen, bezeichnet man als DOS-Attacken. Oftmals handelt es sich um Netzwerkdienste, die beim Trennen der Netzverbindung nicht mehr nutzbar sind. Es werden entweder Schwachstellen im System (Rechner, Übertragungsprotokolle, ...) ausgenutzt oder begrenzte Systemressourcen (z.B. Bandbreite) aufgebraucht, um eine Dienstverweigerung zu erreichen.

Manchmal geht es auch darum, sicherheitsrelevante Systeme (z.B. IDS, Loghosts) in einen instabilen Zustand zu versetzen und somit Einbruchsversuche zu ermöglichen. Prinzipiell werden DOS-Attacken in zwei Arten untergliedert: hostbasierte Denial of Service Attacken und netzwerkbasierte Denial of Service Attacken. Eine besonders gefährliche Weiterentwicklung der letztgenannten sind verteilte DOS-Attacken (Distributed Denial of Service). Das Opfer wird hierbei von mehreren (manchmal sogar tausenden) Rechnern angegriffen.

Hostbasiert

Wenn der Angreifer bereits Zugang zum System besitzt, kann er mit meist sehr einfachen Mitteln das System handlungsunfähig machen. Dazu gehören beispielsweise

* Plattenplatz aufbrauchen (Sicherheitsmaßnahme: `quota`)
* Arbeitsspeicher / CPU Reserven aufbrauchen (Sicherheitsmaßnahme: `ulimit`)
* Hardwarefehler ausnutzen (Sicherheitsmaßnahme: neuer Kernel)

Netzwerkbasiert

Die größere Gefahr geht im Allgemeinen von netzwerkbasierten DOS-Attacken aus. Diese sind schwerer nachvollziehbar, und ein Angreifer benötigt im Zweifelsfall nicht einmal einen direkten Zugang zum Opfernetzwerk. Zu diesen DOS-Attacken gehören:

Email-Bomben

Die Speicherkapazität auf Mailservern ist sehr begrenzt, der Plattenplatz wird sehr schnell aufgebraucht. Außerdem können durch sehr große Mails die Netzwerkressourcen (Bandbreite) sehr schnell aufgebraucht werden.

Broadcast-Angriff

Ein Angreifer generiert pro Sekunde 100 Brodcast-Anfragen. Wenn angenommen 150 Rechner im Subnetz aktiv sind und deshalb antworten, müssen 15.000 Antwortpakete übertragen werden. Der Rechner, der diese Anfrage gestellt hat, ist mit den Antworten hoffnungslos überlastet und verliert im Extremfall seine Netzverbindung (meist in Kombination mit Spoofing). Gegenmaßnahmen können beispielsweise durch die Limit Option im Paketfilter oder durch das Herausfiltern von ICMP Paketen realisiert werden.

Kernel-Angriff

Auch die Netzwerkmodule des Kernels sind nicht vor Angriffen gefeit. So kann beispielsweise durch eine SYN-Flood-Attacke der Netzwerkpuffer zum Überlaufen gebracht werden. Kernelparameter bieten einen guten, meist ausreichenden Schutz.

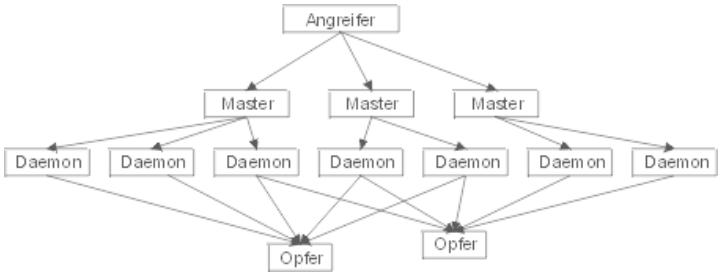
DOS-Angriffe auf TCP/IP-Ebene

Durch Massenanfragen werden Schwachstellen im IP-Stack ausgenutzt. Die Schwachstellen lassen sich nur bedingt beseitigen, sodass solche Angriffe meist als DDOS Attacken durchgeführt werden. Zu solchen Attacken zählen:

* IP-Fragmentangriff (teardrop, newtear, bonk)
* IP-Bombing (meist als DDOS-Attacke)
* SYN-Flood Attacken (massenhafte Verbindungsanfragen)

3.11 Distributed Denial of Service (DDOS Attacken)

Während bei **normalen** netzwerkbasierten DOS-Attacken viele Anfragen von *einem* Angriffsrechner ausgehen, kommen bei verteilten DOS-Attacken (Distributed Denial of Service = DDOS) sehr viele Computer zum Einsatz, die simultan einen DOS-Angriff auf einen Opferrechner durchführen. Ein DDOS-Angriff wird zentral kontrolliert, es gibt also einen oder mehrere Server (so genannte Master), die viele Clients (auch einfach als Daemons bezeichnet) kontrollieren.



ddos.png

Diese Angriffe sind erst seit 1999 bekannt. Die dokumentierten Angriffe gingen meistens von vier Softwaretools aus:

* trinoo (englische Infos auf CERTwww.cert.org/incident_notes/IN-99-07.html)
* TFN Tribe Flood Network (englische Infos auf CERT)
* stacheldraht (basiert auf trinoo und TFN)
* shaft

Die zu ergreifenden Schutzmaßnahmen sind

* Einbruchserkennungssysteme (IDS)
* Paketfilter (ICMP-Filterung, Pakete > 9 Byte)
* Verhinderung eines Einbruchs durch Kombination von Sicherheitsmaßnahmen wie kontinuierliche Aktualisierung der Software, Einsatz von Proxies, Paketfiltern, Einbruchserkennung und Gegenmaßnahmen bei erfolgreichem Angriff.

3.12 Root Kits

Root Kits sind Softwarepakete, in denen meist skriptbasierte Werkzeuge enthalten sind. Diese haben zum Ziel,

1. verschiedene Angriffsformen anzubieten,
2. Angriffe zu verschleiern sowie
3. Hintertüren einzubauen.

Ein solcher automatisierter Angriff dauert typischerweise drei bis fünf Sekunden und erfordert keinerlei Fachwissen.

Wie läuft ein Angriff mit einem Root Kit ab?
Nach dem Ausnutzen einer Schwachstelle, durch die ein beliebiger Maschinencode zur Ausführung gebracht werden kann, steht dem Angreifer eine Ausführungsumgebung mit root-Rechten (daher der Name) zur Verfügung. Diese Privilegien werden benutzt, um auf dem angegriffenen System eine Hintertür zu installieren, die dem Angreifer in Zukunft jederzeit Administratorrechte verschafft. Abschließend werden sämtliche Spuren des Angriffs aus den Log-Dateien entfernt. Präventive Maßnahmen sollten bevorzugt zum Einsatz kommen, da erfolgreiche Einbrüche äußerst schwer zu finden und zu beseitigen sind. Bekannte Root Kits können mit chkrootkit gefunden werden.

4 Allgemeine Schutzmaßnahmen

An dieser Stelle sollen nun ganz grundsätzliche Sicherungsmöglichkeiten aufgezeigt werden, die ohne komplexe Systemeingriffe nachvollzogen werden können. Sicherlich ist der Einsatzzweck des Rechners (Desktop-System, Mail-Server, Firewall, ...) relevant für die Auswahl geeigneter Methoden, deshalb können die hier vorgestellten Maßnahmen weder ausreichend noch vollständig sein!

4.1 Absicherung des Boot-Vorgangs (System-V-Init-Process)

Zuerst sollten die physischen Zugänge (Disketten, CD-ROM, USB/Netzwerk-Boot) gesichert werden. Das kann man ganz einfach im BIOS erledigen. (Passwortschutz für das BIOS ist ebenfalls zu aktivieren!) Einziges **physisches Restrisiko** bleibt der Ausbau der festplatte. Damit ist man allerdings noch längst nicht am Ende...

Linux Systeme lassen sich im Single User Modus starten, indem man z.B. beim LILO-Bootmanager die Option single angibt:

lilo: `linux single`

Im ungünstigsten Fall benötigt man nicht einmal ein Passwort und erlangt Superuser-Rechte. Diese Schwachstelle ist folgendermaßen zu beseitigen:

* Editieren der Datei /etc/inittab

/etc/inittab

```
....
id:3:initdefault:
~~:S:wait:/sbin/sulogin
....
```

* Editieren der Datei /etc/lilo.conf

/etc/lilo.conf

```
....
prompt
timeout=00
default=linux
restricted
passwort=GEHEIM

image=/boot/vmlinuz-2.4.20
    label=linux
....
```

* bei Auswahlmöglichkeit aus mehreren Betriebssystemen z.B. Windows, Linux den Parameter `timeout` auf 50 setzen
* Zugriffsrechte setzen (`chmod 600 /etc/lilo.conf`)
* Die neue Boot-Konfiguration aktivieren (`lilo -v`)
* Die /etc/lilo.conf vor Manipulation schützen (siehe Abschnitt "Erweiterte ext2-Dateiattribute")

Jetzt sollte ein Einbruch durch Ausnutzen der Schwachstellen im Boot-Prozess wesentlich schwieriger zu realisieren sein.

4.2 Nicht benötigte Pakete

Die großen Distributionen *SuSE* und *RedHat* installieren standardmäßig dutzende Pakete, die zum nicht unbedingt notwendig sind und nur zusätzliche Funktionen bieten. Da sichere Systeme immer nach dem Prinzip des Minimalismus zu entwerfen sind, sollten diese Pakte deinstalliert werden. Solche Pakete sind bei *RedHat* z.B.

`apmd`, `anacron`, `at`, `bc`, `eject`, `getty_ps`, `gd`, `gpm`, `isapnptools`, `kernel-pcmcia-cs`, `kudzu`, `linuxconf`, `mailcap`, `mouseconfig`, `mt-st`, `pump`, `pciutils`, `raidtools`, `redhat-logos`, `redhat-release`, `setserial`, `XFree86-SVGA`

Hinweis: Natürlich beinhaltet jedes dieser Pakete sinnvolle Programme, und möglicherweise benötigen Sie sogar eines von den oben angegebenen.

4.3 Nicht benötigte Benutzer und Gruppen

Die meisten Distributionen legen nicht benötigte Nutzer und Gruppen an bzw. vergessen beim Deinstallieren von Softwarepaketen (z.B. `apache-http-server`) das Entfernen der Einträge in den Passwortdateien.

Solche Kennungen sind zum Beispiel: `adm`, `lp`, `sync`, `mail` (bei `sendmail` benötigt!), `ftp` (bei anonymous ftp benötigt),... Folgende Gruppen sind unter Umständen obsolet: `adm`, `lp`, `news`, `pppusers`, `popusers`.

Man sieht, dass solche Benutzer manchmal doch benötigt werden.

4.4 Differenzierter Superuser-Zugriff (root)

Die Sicherheit des root-Kennworts ist unbedingt an den Regeln im Kapitel sichere Passwörter auszurichten.

Viele Programme benötigen nicht unbedingt root-Rechte. Oftmals reicht es aus, eine privilegierte Gruppe anzulegen und mittels der Gruppenrechte den Zugriff zu ermöglichen.

Wenn zum Beispiel das Wechseln der Benutzerkennung nur für bestimmte Benutzer erlaubt werden soll, legt man eine neue Gruppe "security" an und ändert die Zugriffsrechte des Programms `su`.

```
root@linux / # groupadd security
root@linux / # chgrp security /bin/su
root@linux / # chmod 4750 /bin/su
```

Feiner abgestimmte Restriktionen bieten Capability-Mechanismen des Kernels, die den root-Zugriff weiter einschränken.

4.5 Capability Mechanismus des Kernels

Der Entzug von Capabilities (Systemfähigkeiten) erlaubt bei Systemstart bestimmte Restriktionen durchzusetzen und ermöglicht damit unter anderem eine Beschränkung der Superuser-Rechte. Nach dem Boot-Vorgang sind standardmäßig alle Capabilities gesetzt.

Der Capability Mechanismus wurde in POSIX 1003.1e und IEEE 1003.2c beschrieben und festgelegt.

Folgende Übersicht beschreibt ganz kurz einige Capabilities (Systemfähigkeiten):

CAP_LINUX_IMMUTABLE	Ändern von Dateiberechtigungen (siehe Abschnitt Erweiterte ext2-Dateiattribute)
CAP_SYS_RAWIO	direkten Zugriff auf Speichermedien (Festplatten)
CAP_SETGID	setgid setzen
CAP_SETUID	suid setzen
CAP_SYS_MODULE	Kernelmodule laden
CAP_SYS_ADMIN	mounten/unmounten und andere administrative Tätigkeiten
CAP_NET_ADMIN	Netzwerkkarten konfigurieren, Firewall administrieren, Routing administrieren

Mittels des Programmes `lcap` können dem System einzelne Fähigkeiten entzogen werden:

lcap CAP_LINUX_IMMUTABLE
lcap CAP_SYS_RAWIO

Nun ist es auch für root nicht mehr möglich, durch `+i` oder `+a` geschütze Dateien im ext2 Dateisystem zu löschen. (siehe Abschnitt Erweiterte ext2-Dateiattribute)

An dieser Stelle sei darauf hingeweisen, dass CAP_SYS_RAWIO unbedingt sehr frühzeitig entfernt werden sollte, da ansonsten mittels direktem Zugriff auf das /dev/kmem device der Capability Schutz umgangen werden kann.

4.6 Pfadvariable PATH

Die Shell-Variable `PATH` beinhaltet eine Liste an Verzeichnissen, die beim Aufruf eines Programmes durchsucht werden.

```
user@linux / $ echo $PATH

/usr/local/bin:/usr/bin:/bin:/usr/bin/X11
```

Die Verzeichnisse werden mit einem Doppelpunkt getrennt. Aus Bequemlichkeit beinhaltet die Pfadvariable oftmals das aktuelle Verzeichnis. Es ist darauf zu achten, dass dieses immer am Ende steht (Bedrohung: Hackerprogramme in allgemein zugänglichen Verzeichnissen wie z.B. /tmp oder /usr/share/firmaXYZ tarnen sich als `su` oder `passwd`)

```
user@linux / $ echo $PATH

/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:.
```

4.7 Namensauflösung

In der Datei /etc/hosts.conf wird festgelegt, WIE die Zuordnung der Namen zu IP-Adressen erfolgen soll. Das Beispiel zeigt eine relativ sichere Variante.

/etc/hosts.conf

```
# Reihenfolge der Domain Namens Auflösung
#     (vorausgesetzt DNS/BIND verfügbar)
order hosts, bind
# Multiple IP Adressen werden nur bei Routern benötigt
multi off
#Anti IP-Spoofing
nospoof on
```

4.8 Festlegung vertrauenswürdiger Rechner / Netzwerke

Aus Sicherheitsgründen ist von einer Benutzung der **r-Dienste** (z.B. `rsh`, `rcp`, `rexec`) unbedingt abzuraten. Demzufolge sollten auch keine .rhosts Dateien auf dem Rechner zu finden sein, dies kann man folgendermaßen überprüfen:

```
user@linux / $ find /home -name .rhosts
```

Unbedingt sollte man sich mit allen Möglichkeiten von `ssh` vertraut machen, insbesondere mit dem Umgang von Schlüsselpaaren. Ein guter Einstieg in dieses Thema ist auf den Web Seiten des Linux-Magazins zu finden.

4.9 Festlegung (nicht) benötigter Protokolle (/etc/services)

In der Datei /etc/services werden alle Dienste ihren Ports zugewiesen. Eine Manipulation kann einem Angreifer Hintertürchen öffnen und sollte deshalb unbedingt unterbunden werden (siehe Abschnitt Erweiterte ext2-Dateiattribute).

4.10 Verwaiste und rechtelose Dateien

Dateien, die von jedermann beschrieben werden können, stellen ein sehr großes Sicherheitsrisiko dar. Deshalb sollten so wenig wie möglich solcher Dateien existieren (Gruppenrechte einsetzen!). Mit `find` lassen sich solche Dateien aufspüren und nach sorgfältiger Kontrolle kontrolliert löschen:

```
root@linux / # find / -perm -o+w -a ! -type l -ls >/tmp/rechtlose-files
```

Verwaiste Dateien gehören keinem Nutzer oder keiner Benutzergruppe an. Dies deutet auf einen erfolgreichen Einbruch hin und sollte kontinuierlich untersucht werden:

```
root@linux / # find / -nouser -o -nogroup >/tmp/verwaiste_files
```

4.11 shadow Passwörter

Bei Verwendung der Shadow Suite werden Passwörter verschlüsselt in der Datei /etc/shadow abgelegt. Diese kann nur vom Superuser root und von der Gruppe **shadow** gelesen werden. Kein realer Benutzer hat also Zugang und kann die Passwörter lesen und versuchen, diese mit Hilfe eines Crack-Programmes (z.B. John the Ripper) zu entschlüsseln. Empfehlenswert ist das Abspeichern der Passwörter als MD5-Hash, da dadurch ein potentieller Angriff schwerer durchzuführen ist und längere Passwörter verwendet werden können. Weiterführende

Informationen sind im Abschnitt Authentisierung, Autorisierung und Zugriffssteuerung mit PAM zu finden.

4.12 Schutz der Protokolldateien (Systemlog-Files)

Eine einfache Absicherung bieten die erweiterten Dateiattribute wie beispielsweise das append-only Attribut bei ext2-Filesystemen (siehe Abschnitt Erweiterte ext2-Dateiattribute). Eine Alternative dazu besteht darin, die Logfiles auf einem dafür bestimmten Rechner zu sammeln. In diesem Zusammenhang sei auf den `ssyslogd` und den `syslog-ng` verwiesen, die zusätzlich Integrität und Verschlüsselung bieten.

4.13 Bastille Linux

Diese Sammlung von Perl-Skripten erhöht die Sicherheit durch eine automatische Änderung der Konfigurationseinstellungen. Vor jeder Modifikation wird man sehr ausführlich über die Konsequenzen informiert und kann nach dem Abschätzen des Für und Wider die Änderungen akzeptieren oder auch nicht. Das System ist kostenlos und kann unter http://bastille-linux.org heruntergeladen werden. Das Ausführen sollte im Single-User Modus erfolgen:

```
root@linux / # mv Bastille-XXX.tar.gz /root
root@linux / # cd /root
root@linux / # tar zxvpf Bastille-XXX.tar.gz
root@linux / # init 1
root@linux / # cd /root/run-Bastille
root@linux / # ./Bastille1.pl
```

Zum Schluss noch ein Zitat, auf das mich *Steffen Dettmer* hingewiesen hat:

> "The only secure computer system in the world is unplugged, locked in a vault at the bottom of the ocean and only one person knows the location and combination of that vault. And he is dead."
> --Bruce Schneier in "Applied Cryptography"

5 Authentisierung, Autorisierung und Zugriffssteuerung mit PAM

Die meisten Linux-Systeme sind von anderen Netzwerkrechnern über verschiedene Zugänge wie beispielsweise `ssh`, `ftp` oder `pop3` erreichbar. Bevor ein solcher Zugang gewährt wird, muss die Identität des Benutzers zweifelsfrei festgestellt werden (Authentisierung). Jeder Zugangsdienst, also beispielsweise der FTP Server, muss nun diese Überprüfung selbst implementieren (i.d.R. Passwort).

Soll aber nun zum Beispiel ein Zugang nur in festgelegten Zeitfenstern möglich sein oder eine Chipkarte bzw. ein biometrisches Verfahren (Fingerabdruck, IRIS) das herkömmliche Passwortverfahren ersetzen, so müsste man alle Zugangsprogramme (`login`, `ftp`, `ssh`, ...) dementsprechend modifizieren. Ebenso schwierig gestaltet sich die Integration in Sicherheitsarchitekturen wie *DCE* oder *Kerberos*.

Bereits Anfang der 90er Jahre erkannte Sun Microsystems diese Nachteile und entwickelte und implementierte den so genannten PAM-Standard in ihrem Betriebssystem Solaris. Seit Oktober 1995 gibt es den RFC 86.0, der PAM allgemein beschreibt. Was ist nun PAM?

PAM steht für **P**luggable **A**uthentication **M**odules was auf deutsch heißt: steckbare Authentisierungsmodule. Es handelt sich also um ein modulares System, welches die Anwendungen (`login`, `ssh`, `ftp`) von den Mechanismen zur Benutzerauthentisierung trennt. Einzelne Authentifizierungsschritte können nacheinander durchgeführt werden, ohne dass der Benutzer davon etwas merkt.

Für jede Anwendung (z.B. `login`) wird festgelegt, *welche* Sicherheitsmodule *wie* durchlaufen werden. Dazu existieren jeweils eigenständige Konfigurationsdateien im Verzeichnis /etc/pam.d/ :

```
user@linux / $ ls /etc/pam.d

chfn   cron   kdm   passwd   ppp   su   chsh   cvs   login   other   ssh   wu-ftpd
```

/etc/pam.d/login

```
# Modul auth: root kann sich nur an bestimmten
# Konsolen anmelden (definiert in /etc/securetty)
auth       required   /lib/security/pam_securetty.so

# komplexe Passwörter erforderlich
#   (mind. 8 Zeichen, max 3 Login-Versuche)
password   required   /lib/security/pam_cracklib.so retry=3 minlen=8
[...]

# Zeitgesteuerte Zugangsberechtigungen
#   (definiert in /etc/security/time.conf)
account requisite pam_time.so
[...]
```

5.1 Modultypen und Kontroll-Flags

Jeder Eintrag einer solchen Konfigurationsdatei besteht aus mindestens drei Elementen. Es beginnt mit einem *Modultyp* gefolgt von einem *Kontroll-Flag*. An dritter Stelle steht der Name des *Moduls* und optional dahinter weitere Argumente. Das Zusammenspiel

Modultyp - Kontroll-Flag - Modul

wird am Ende des Abschnittes in einem komplexen Beispiel verdeutlicht werden.

Jedes Modul kann einen oder mehrere **Modultypen** implementieren:

Modultyp	Beschreibung	unterstützte Funktion	Beschreibung
`auth`	Authentifizierungsmanagement	pam_authenticate() pam_setcred()	Benutzerauthentisierung Setzen, Erneuern oder Löschen von Berechtigungen
`account`	Zugangsmanagement	pam_acct_mgmt()	Entscheidung, ob Zugang gestattet oder verweigert wird (z.B. zeitliches Zugangslimit)
`session`	Sitzungsmanagement	pam_open_session() pam_close_session()	z.B. Dauer einer Sitzung kontrollieren /protokollieren
`password`	Passwort-Management	pam_chauthtok()	Kontrolle der Passwortänderung

Wie und in welcher Weise die aufgeführten Funktionen eines Moduls relevant sind, kontrollieren die sogenannten **Kontroll-Flags**:

control-flag	Beschreibung
`required`	Modul muss zwingend durchlaufen werden.Auch bei nicht erfolgreichem Durchlaufen werden alle folgenden Module bearbeitet (damit ist nicht ersichtlich, welches Modul den Zugriff verweigert).
`requisite`	Bei einem Fehler (nicht erfolgreiches Durchlaufen) wird sofort zum Anwendungsprogramm (z.B. `login`) zurückgekehrt, ohne die folgenden Module abzuarbeiten. Ansonsten wird in der Abarbeitung der Kette fortgefahren.
`optional`	Bei Erfolg werden trotzdem alle nachfolgenden Module abgearbeitet.
`sufficient`	Bei Erfolg des Moduls wird sofort zur Anwendung zurückgekehrt, die Abarbeitung der anderen Module ist nicht notwendig.

5.2 Erläuterung eines Konfigurationsbeispiels

Die folgende Konfiguration bezieht sich auf das Login und wird deshalb in der Datei /etc/pam.d/login gespeichert. (Die angegebenen Zeilennummern sind KEIN Bestandteil der Datei und dienen nur der besseren Orientierung.)

/etc/pam.d/login

```
 1 # root kann sich nur an Konsolen aus /etc/securetty anmelden
 2 auth     required   /lib/security/pam_securetty.so
 3
 4 # bei Existenz von /etc/nologin darf sich nur root anmelden
 5 auth     required   /lib/security/pam_nologin.so
 6
 7 # Anmelden über Verzeichnisdienst LDAP
 8 auth     sufficient /lib/security/pam_ldap.so
 9
10 # Anmeldung lokaler Benutzer
11 auth     required   /lib/security/pam_unix.so use_first_pass
12
13 # Zeitgesteuerte Zugangsberechtigungen (definiert in /etc/security/time.conf)
14 #account requisite pam_time.so
15
16 # Vergabe der Zugangsberechtigungen entsprechend der LDAP Einträge
17 account  sufficient /lib/security/pam_ldap.so
18
19 # Standard Zugangsberechtigungen
20 account  required   /lib/security/pam_unix.so
21
22 # Passworteinschränkungen bei Änderung
23 password required   /lib/security/pam_cracklib.so minlen=5 dcredit=2 ocredit=2
24
25 # Benutze Shadow Suite und md5 Passwörter
26 password required   /lib/security/pam_unix.so use_first_pass md5 shadow
27
28 # Weitergabe des Passwortes an LDAP
29 password sufficient /lib/security/pam_ldap.so use_authtok
30
31 # generiere Fehler bei nicht erfolgreicher Passwortänderung
32 password required   /lib/security/pam_deny.so
33
34 # Standard Sitzung
35 session  required   /lib/security/pam_unix.so
36
37 # Standard Sitzung
38 session  optional   /lib/security/pam_ldap.so
39
40 # Limits (definiert in /etc/security/limits.conf)
41 # session required pam_limits.so
```

Authentifizierungsmanagement

In Zeile 2 überprüft das Modul `pam_securetty`, ob sich der Superuser root an der Konsole anmelden darf oder nicht. In der dazugehörenden Konfigurationsdatei /etc/securetty wird untereinander angegeben, wo dies möglich sein soll:

#tty1
#tty2
#tty3
#tty4
tty5
#tty6

Da im obigen Beispiel alle Einträge außer tty5 auskommentiert wurden, kann sich root nur an dieser einen Konsole anmelden. In Zeile 5 wird das Modul `pam_nologin.so` durchlaufen, welches auf dem System nach der Datei /etc/nologin sucht. Bei Vorhandensein wird lediglich dem Superuser root das Anmelden gestattet. Zeile 8 integriert das Anmelden über LDAP mittels des Moduls `pam_ldap.so`. LDAP ist ein Verzeichnisdienst, der in einem eigenständigen Selflinux-Kapitel näher beschrieben wurde. Sind

Benutzerkennung und Passwort korrekt, so wird zum `login`-Programm zurückgekehrt. In Zeile 11 wird das Authentisierungsmodul `pam_unix.so` mit dem Passwort des LDAP-Moduls durchlaufen (`use_first_pass.so`). Somit können sich auch lokale Benutzer, die nicht im LDAP-Baum gelistet sind, am System anmelden.

Zugangsmanagement

Zeile 14: Zeitgesteuerte Berechtigungen werden mit dem Modul `pam_time` realisiert. **Achtung!** Um so näher das Ende eines Projektes naht, desto länger arbeiten auch die Mitarbeiter. In Zeile 17 und 20 werden die Zugangsberechtigungen vergeben. Dabei wertet das `ldap.so` Modul verschiedene Parameter aus dem Verzeichnisbaum aus (z.B. `LDAP_ATTRNAME_EXPIRATIONDATE`). Sollte das ldap Modul nicht erfolgreich durchlaufen werden, so ist für die Autorisierung das `pam_unix` Modul verantwortlich.

Passwortmanagement

In Zeile 22-32 werden Passwortänderungen kontrolliert. Das erste Modul `pam_cracklib` untersucht die Passwörter auf ihre Einfachheit. Die Optionen geben an, dass ein Passwort aus mindestens 5 Zeichen bestehen muss, wobei mindestens zwei Ziffern und zwei Sonderzeichen dabei sind. Das nächste Modul (`pam_unix`) übernimmt das Passwort vom `pam_cracklib` Modul und verwendet einen md5 Hash und die Passwort-Shadow-Suite. Anschließend wird das Passwort an das ldap Modul weitergereicht, und falls dieses es nicht erfolgreich verarbeiten kann, wird das `deny`-Modul in Zeile 31 einen Fehler generieren. Wozu muss aber überhaupt der `login`-Prozess das Passwort ändern können? Wie im Abschnitt zuvor angesprochen, können Passwörter verfallen, und dann wird der Benutzer beim Anmelden (also beim `login`-Prozess) aufgefordert, sein Passwort zu ändern.

Sitzungsmanagement

Im letzten Abschnitt befinden sich die Module für das Sitzungsmanagement. `Pam_unix` protokolliert in den Logfiles nicht nur den Beginn einer Sitzung sondern auch deren Ende und weitere sicherheitsrelevante Informationen. Das ldap Modul ist optional, das heißt ein erfolgreiches Durchlaufen ist nicht unbedingt erforderlich. Zum Schluss wird das Modul `limits` eingebunden, welches die Systemressourcen wie beispielsweise Hauptspeicherbedarf, CPU-Zeit, Prozesse und Dateien für einzelne Benutzer bzw. Benutzergruppen begrenzt. Die Einstellungen dazu werden in der Konfigurationsdatei /etc/security/limits.conf vorgenommen:

/etc/security/limits.conf

```
#/etc/security/limits.conf
# Maximal 4 Nutzer der Benutzergruppe "user" dürfen sich anmelden
@users    -     maxlogins     4
# Prozesse des Benutzers "www" werden mit nice-Level 17 ausgeführt
www     -       priority       17
# Prozesse der Gruppe "users" werden mit nice-Level 17 ausgeführt,
# Priorität kann aber durch den Benutzer geändert werden
@users    soft       priority       17
# root darf nur einmal angemeldet sein
root       hard    maxlogins     1
```

5.3 weitere wichtige PAM-Module

`pam_access`	eingeschränkte Zugangsberechtigungen (Benutzer <--> Terminal) Konfigurationsdatei: /etc/security/access.conf
`pam_group`	Zuweisung zu Benutzergruppen entsprechend bestimmter Kriterien (Benutzer,

	Zeit, Terminal). Konfigurationsdatei: /etc/security/group.conf
pam_rhosts_auth.so	Unterbindung des Remote Zugriffs durch .rhosts Dateien
pam_console.so	Räumt normalen Benutzern zusätzliche Rechte ein.(ACHTUNG: Dieses Modul sollte in abzusichernden Systemen NIE benutzt werden)
pam_tally.so	Sperrung des Zugangs nach x fehlgeschlagenen Login-Versuchen
pam_wheel.so	Begrenzung des su Kommandos auf eine kleine Benutzergruppe
xpam_time.so	zeitgesteuerte Zugangskontrolle
pam_env.so	nicht veränderliche Umgebungsvariablen
pam_pwdb.so	Authentifikation
mod_auth_pam	PAM Modul für apache Webserver
pam_ftp.so	Anonymous Zugriff (testet Passwort auf "@"-Zeichen)
pam_php.so	PAM Unterstützung für PHP
pam_krb5.so	Kerberos Modul
pam_radius.so	Radius=Remote Authentication Dial-In User Service

Ein kurzer Hinweis zum Schluss: Das Modul pam_console.so sollte in allen Konfigurationsdateien auskommentiert werden. Dies übernimmt das folgende Skript:

```
#!/bin/sh
cd /etc/pam.d
for i in *; do
  sed ´/[^#].*pam_console.so/s/^/#/´ < $i >foo && mv foo $i
done
rm -rf /etc/security/console.apps/*
```

6 Superserver xinetd

6.1 Der Superserver inetd

Superserver sind für die Verwaltung vieler verschiedener Dienste (FTP, POP3, Telnet, Finger) zuständig. Erst wenn ein Dienst benötigt wird, startet der Superserver ein zugeordnetes Programm. Im Falle von FTP sieht das folgendermaßen aus: In der Konfigurationsdatei /etc/inetd.conf wird der FTP-Port dem FTP-Daemon zugewiesen. Wenn jemand versucht, eine Verbindung zu diesem Port aufzubauen, startet der Superserver den zugewiesenen FTP-Daemon. (Hinweis: Die Zuweisung der Port-Nummern erfolgt in /etc/services)

6.2 Zugriffssteuerung mit TCP_Wrapper

Die Nutzung von Serverdiensten sollte aus Sicherheitsgründen kontrolliert erfolgen. Bestimmte Dienste wie NFS oder Samba sollten nur im Intranet bzw. nur in bestimmten Netzwerkbereichen zur Verfügung stehen. Diese Kontrollfunktion kann der TCP_Wrapper `tcpd` übernehmen. Wie kann man sich das nun vorstellen? Normalerweise sieht ein Eintrag in der Konfigurationsdatei des Superservers /etc/inetd folgendermaßen aus:

/etc/inetd

```
sshd stream tcp nowait root /usr/sbin/sshd sshd -i
```

Eine Verbindungsanfrage zu dem ssh-Port soll angenommen werden und nach erfolgreichem Start des Programms `/usr/sbin/sshd` wird der TCP-Datenstrom an diesen Server weitergeleitet. Der Serverprozess läuft unter root-Rechten ab.

Wenn man kontrollieren möchte, wer den ssh-Dienst ausführen darf, muß man folgende Änderung vornehmen:

/etc/inetd

```
ssh stream tcp nowait root    /usr/sbin/tcpd sshd
```

Nun wird anstelle des ssh-Servers der TCP_Wrapper `tcpd` aufgerufen, und erst wenn dieser die Verbindungsanfrage positiv entscheidet, wird diese an den ssh-Dienst weitergegeben. In den Dateien /etc/hosts.`allow` und /etc/hosts.deny können Netzwerkbereiche für bestimmte Dienste freigeschaltet oder abgeschottet werden. Die folgenden Einträge zeigen einige Konfigurationsmöglichkeiten:

/etc/hosts.allow

```
#/etc/hosts.allow

#Erlaube alle Dienste für das lokale Netz
ALL:ALL:192.168.

# Erlaube SSH für alle außer 192.168.1.5 und alle Rechner von Micrsoft.com
sshd : ALL EXCEPT 192.168.1.5 .microsoft.com : ALLOW

#Erlaube SSH für das lokale Netz
sshd : 192.168.1.0/255.255.255.0 : ALLOW

#Alle bis hierin noch nicht behandelten Anfragen werden an root gemailt
ALL : ALL : (echo "Zugriff von %c auf %s" | mail -s "%d-Zugriff" root)
```

Ein Nachteil des TCP_Wrappers ist der fehlende Schutz vor IP-Spoofing Attacken, in denen eine falsche Identität anderer Rechner vorgetäuscht wird. Ein anderer Nachteil ist, dass Dienste mit dem TCP_Wrapper zusammenarbeiten müssen (Unterstützung der Bibliothek libwrap).

6.3 Die bessere Alternative xinetd

Der Superserver `xinetd` ist eine Weiterentwicklung des `inetd`-Konzeptes mit umfangreichen Sicherheitsoptionen. Es wurden zahlreiche Erweiterungen in folgenden Bereichen vorgenommen:

* Zugriffskontrolle
* Protokollierung
* Dienst-Weiterleitung
* Schutz vor DOS Attacken
* IPv6 Unterstützung

Zugriffskontrolle

Netzwerkanfragen aus nicht vertrauenswürdigen Netzwerkbereichen können aufgrund ihrer Ursprungsadresse abgelehnt werden (ähnlich TCP_Wrapper). Die weitere Nutzung der Dateien /etc/hosts.allow und /etc/hosts.deny ist möglich. Außerdem unterstützt `xinetd` eine Zeitbegrenzung für Dienste (access-time). Es können sowohl die maximalen Netzwerkverbindungen für einen Client (per_source) als auch auch für einen Dienst (instances) limitiert werden.

Log-Funktionen

Die Protokollierung erfolgt entweder mittels `syslogd` oder in separaten Dateien. Im ersten Fall, sind verschiedene Log-Level für jeden einzelnen Netzwerkdienst möglich. Neben Benutzerkennung (userID) und Server-Prozessidentifikation (PID) kann auch die Client-IP (HOST) und die Dauer einer Verbindung protokolliert werden. Ausserdem beinhaltet der `xinetd` umfangreiche Funktionen zur Darstellung fehlgeschlagener Verbindungsanfragen.

Dienst-Weiterleitung

TCP-Datenströme können zu einem anderen Rechner weitergeleitet werden (redirect). Es können auch Dienste weitergeleitet werden, die normalerweise von Außen nicht erreichbar sind. Dies ist eine mögliche Schwachstelle, man sollte dieses Feature daher nur sehr überlegt einsetzen!

Schutz vor DOS Attacken

Durch eine Begrenzung der Verbindungsanzahl verlieren so genannte "PORT-Bomben" ihre Wirkung (per_source, instances).

Wenn die Protokollierung nicht über den `syslogd` sondern in separaten Dateien erfolgt, kann eine Überflutung dieser Dateien durch Begrenzungen verhindert werden (log_type FILE datei soft_limit hard_limit).

IPv6 Unterstützung

Seit Version 2.1.8.8.pre wird das IPv6 Protokoll unterstützt.

Weitergehende Informationen und auch Beispiele findet man im Kapitel xinetd von SelfLinux.

7 Sicherheit im Dateisystem

7.1 Dateiattribute

Grundlage für die folgenden Unterabschnitte sind die Dateiattribute LESEN, SCHREIBEN, AUSFÜHREN und damit verbunden die Berechtigungsklasen USER, GROUP, OTHER. Diese wurden ausführlich im Abschnitt 4 und 5 des Kapitels Benutzer- und Berechtigungskonzepte unter Linux erläutert.

7.2 t-Bit (sticky Bit), SGID, SUID

Das sticky-Bit hat historischen Ursprung und wird heute nur noch auf Verzeichnisse angewendet (Früher blieben mit dem t-Bit versehene Programme auch nach Beendigung im Hauptspeicher).

Ist das Sticky Bit auf einem Verzeichnis z.B. /tmp gesetzt, dann dürfen Dateien nur noch von dessen Besitzer und vom Verzeichnisbesitzer gelöscht werden.

```
root@linux / # chmod 1777 public_tmp
root@linux / # ls -la public_tmp

drwxrwxrwt    7 root     root         4096 2003-07-06 19:17 .
```

SUID und SGID sind nur für Programme sinnvoll. Diese werden mit der Benutzerkennung
(SUID - Set User ID) bzw mit der Gruppenkennung
(SGID - Set Group ID) der Programmdatei ausgeführt.

Anwendung findet dies beisielsweise beim Programm `passwd`, welches ja Schreibrechte auf die Datei /etc/shadow benötigt um das Passwort zu ändern. Das Programm wird also von einem realen Nutzer aufgerufen, läuft aber unter root-Rechten ab:

```
root@linux / # ls -la /usr/bin/passwd

-rwsr-xr-x    1 root  root  24248 2003-04-26 21:50  /usr/bin/passwd
```

User ID bei Ausführung des apache setzen
```
chmod 4755 /usr/bin/passwd
```

#Gruppen ID bei Ausführung von my_script setzen
```
chmod 2755 /usr/bin/my_script
```

Bit-Maske	Bedeutung
4000	User ID setzen (bei Programmen)
2000	Gruppen ID setzen (bei Programmen)
1000	Lösch- und Überschreibschutz aktivieren (Verzeichnisse)
400	Eigentümer lesen
200	Eigentümer schreiben / modifizieren
100	Eigentümer ausführen /Verzeichnis wechseln
40	Gruppe lesen

20	Gruppe schreiben / modifizieren
10	Gruppe ausführen /Verzeichnis wechseln
4	alle lesen
2	alle schreiben / modifizieren
1	alle ausführen /Verzeichnis wechseln

7.3 Erweiterte ext2-Dateiattribute

Seit dem Linux-Kernel 2.2 ist es möglich, für Dateien und Verzeichnisse erweiterte (Sicherheits-)Funktionen durch die Vergabe von Flags (Attributen) zu aktivieren. Diese betreffen auch den Superuser root.

a (Append Only)

Das System erlaubt das Öffnen dieser Datei nur zum Zwecke der Erweiterung und verbietet explizit allen Prozessen das Überschreiben bzw. Löschen bereits gespeicherter Daten (sinnvoll bei log-Dateien). Ebenfalls untersagt ist das Löschen, Umbenennen, Verschieben und "hartes" Linken der Datei. Wird das Flag auf einem Verzeichnis angewendet, dürfen darin keine Dateien angelegt oder gelöscht werden.

i (Immutable)

Das System verbietet jegliche Änderungen an der Datei. Im Falle eines Verzeichnisses können Dateien, die in diesem Verzeichnis bereits existieren, verändert werden. Weder das Löschen noch das Anlegen von Dateien ist möglich.

d (No Dump)

Das dump Programm soll diese Datei beim Backup ignorieren.

c (Compress)

Das System soll diese Datei "transparent" komprimieren. Beim Schreiben in die Datei wird deren Inhalt komprimiert und erst danach auf dem physischen Datenträger abgelegt, beim Lesen der Datei werden jedoch stets dekomprimierte Daten zurückgegeben. Diese Funktion ist derzeit noch nicht implementiert.

s (Secure Delete)

Wenn das System diese Datei löscht, werden alle Datenblöcke auf dem Datenträger zufällig überschrieben.

u (Undelete)

Wenn eine Anwendung das Löschen der Datei fordert, soll das System die Datenblöcke so "konservieren", dass das Löschen der Datei auch wieder rückgängig gemacht werden kann. Diese Funktion ist derzeit noch nicht impementiert.

A (Atime)

Das System soll die "atime" (access time) dieser Datei nicht aktualisieren.

S (Sync)

Wenn eine Anwendung gerade den Schreibprozess durchführt, soll das System alle Änderungen sofort und ungepuffert auf dem physischen Datenträger abspeichern.

```
root@linux / # chattr +a test.log
root@linux / # chattr +i test.conf
root@linux / # ls -al test*

-rw-rw-r--    1 gwe  users           0 Nov 22 22:22 test.conf

root@linux / # lsattr -a test*

---i-------- test.conf
----a------- test.log

root@linux / # chattr -i test.conf lsattr -a test.conf

------------ test.conf
```

In vielen Fällen ist der Schutz der folgenden Dateien/Verzeichnisse sinnvoll:

```
root@linux / # chattr -R +i /etc /bin /sbin /boot /lib
root@linux / # chattr -R +i /usr/bin /usr/sbin /usr/lib /usr/src
/usr/include
root@linux / # chattr +a /var/log/messages /var/log/secure
root@linux / # (...)
```

Der Schutz folgender Verzeichnissen kann zu unerwünschten Nebeneffekten führen:

Verzeichnis	Problemursache
/	syslog
/dev	Syslog will beim Start den Socket /dev/log anlegen
/tmp	versteht sich von selbst
/var	logrotate, sendmail

Mithilfe der oben beschriebenen Capabilities kann das Rücksetzen der Sicherheitsattribute unterbunden werden. Die damit realisierten Restriktionen gelten für alle Benutzer (inkl. root) und werden nur bei Systemsneustart aufgehoben.

7.4 Partitionen

Das Einbinden von Partitionen kann sehr viel zur Sicherheit beitragen. Als Beispiel soll der folgende Eintrag in der /etc/fstab dienen:

/etc/fstab

```
/dev/hda9    /tmp    ext2    defaults,nosuid,noexec,nodev    0    2
```

Aufgrund der Mount-Optionen werden gesetzte SUID- und GUID-Attribute komplett ignoriert. Weiterhin verbietet `noexec` das Ausführen von Programmen auf dieser Partition und `nodev` verhindert character und block-Devices.

Die folgenden Einträge geben Überblick über eine mögliche Konfiguration:

/etc/fstab

```
/dev/sda5   /usr         ext3    defaults,ro,nodev                           0   2
/dev/sda7   /usr/share   ext3    defaults,ro,nodev,nosuid                    0   2
/dev/sda8   /var         ext3    defaults,nodev,usrquota,grpquota            0   2
/dev/sda9   /tmp         ext3    defaults,nodev,nosuid,noexec,usrquota,grpquota 0   2
/dev/sda10  /home        ext3    nosuid,nodev,auto,nouser,usrquota,grpquota  0   2
/dev/hda1   /mnt/hda1    vfat    rw,nosuid,nodev,umask=000,uid=1000,gid=1000 0   0
/dev/fd0    /mnt/fd0     ext2    defaults,users,nodev,nosuid,noexec          0   0
/dev/hdd    /mnt/cdrom   iso9660 ro,users,nodev,nosuid,noexec                0   0
```

Die Optionen sind im Einzelnen in den man-Pages (`man 8 mount`) nachzulesen, deshalb gibt es hier nur eine kleine Auswahl:

defaults	Entspricht den voreingestellten Optionen rw, suid, dev, exec, auto, nouser, und async.
rw	Einhängen des Dateisystems zum Lesen und Schreiben.
ro	Einhängen des Dateisystems ausschließlich zum Lesen, Schreiboperationen werden ignoriert.
user	Ein Normaluser darf das Dateisystem einhängen. Ansonsten darf dies nur root.
nouser	Nur root darf das Dateisystem einhängen.
dev / nodev	Das Nutzen von Gerätedateien auf der Partition ist (nicht) erlaubt.
exec / noexec	Auf der Partition gespeicherte Programme können (nicht) ausgeführt werden.
auto / noauto	Partition wird beim Booten (nicht) automatisch eingehängt.
atime / noatime	Sie Zugriffszeit (atime) wird bei jedem Zugriff (nicht) gesetzt.
suid / nosuid	SUID und SGID Bits werden (nicht) interpretiert.
sync / async	Ein- und Ausgabeoperationen werden (a)synchron durchgeführt

Jedes Dateisystem stellt selbst noch einige Optionen zur Verfügung (z. B. usrquota, gid, uid, umask). Die Dokumentation dieser speziellen Optionen kann ebenfalls in den Manpages von `mount` nachgelesen werden.

7.5 Festplattenkontingente mit Quota

Jeder Computernutzer weiß, dass eine Festplatte, egal wie groß sie auch sein mag, immer zu klein ist. Auf Systemen, die von mehreren Personen genutzt werden, wird dies sehr oft zum Problem. Damit es gar nicht erst zu einem Streit kommt, kann der Administrator mit root-Rechten für jeden Nutzer bzw. für jede Nutzergruppe ein Kontingent (engl. Quota) an Speicherplatz zur Verfügung stellen. Dazu benötigt man Quota.

Die Kontingente werden nicht global für das ganze System, sondern für jede Partition separat vergeben. Wenn ein Nutzer Schreibrechte auf mehreren Partitionen besitzt, muss man auch mehrere Quota für diesen Benutzer setzen.

Man kann sowohl Quotas auf Benutzerebene als auch auf Gruppenebene festlegen. Bei der Kombination von Benutzerebene und Gruppenebene haben Quotas auf Benutzerebene eine höhere Priorität.

Mit den folgenden drei Parametern lassen sich Quotas einstellen:

Softlimit	Der Benutzer darf diese Grenze nur kurz überschreiten und wird extra gewarnt.
Grace Period	Dieser Parameter legt den Zeitraum fest, den ein Benutzer das Softlimit überschreiten darf.
Hardlimit	Diese Grenze darf nicht überschritten werden.

Bevor man Quotas festlegen kann, muss der Kernel mit der Option `Quota Support` kompilert werden. Weiterhin ist es erforderlich, die Dateisysteme darauf vorzubereiten (`mount` Option). Dazu wird die Datei /etc/fstab angepasst:

/etc/fstab

```
...
/dev/sda7   /home  ext3    nosuid,nodev,auto,nouser,usrquota,grpquota  0   2
...
```

Nun müssen die Änderungen übernommen werden. Dazu ist nicht unbedingt ein reboot erforderlich es reicht durchaus, die Partition aus- und wieder einzuhängen.

```
root@linux / # mount -a -o remount
```

Weiterhin müssen die Dateien aquota.user und aquota.group im Wurzelverzeichnis der jeweiligen Partition erstellt werden. Dies übernimmt auch der erstmalige Aufruf des Programmes `quotacheck`. Damit die Benutzer auch den aktuellen Stand ihres Kontingentes erfragen können, ist der Lesezugriff auf diese quota-Dateien notwendig.

```
root@linux / # quotacheck -avugm
root@linux / # chmod 644 /home/aquota.*
root@linux / # chmod 644 /moutpoint/aquota.*
```

Das Starten und Beenden der Kontingentüberwachung übernehmen die Programme `quotaon` und `quotaoff`, die nach jedem Systemstart neu aufgerufen werden müssen. Wenn bei der Distribution ein entsprechendes SysV-Initskript fehlt, kann man folgendes benutzen:

/etc/init.d/quota

```
#! /bin/sh

case "$1" in
    start)
        echo "Starte Quotaüberwachung:"
        /sbin/quotaon -avug
        ;;
    stop)
        echo "Stoppe Quotaüberwachung:"
        /sbin/quotaoff -avug
        ;;
    *)
        echo "Aufruf: $0 {start|stop}"
        exit 1
esac
exit 0
```

```
root@linux / # cd /etc/init.d
root@linux / # chmod +x quota
root@linux / # ln -s ../init.d/quota /etc/rc0.d/K85quota
root@linux / # ln -s ../init.d/quota /etc/rc1.d/K85quota
root@linux / # ln -s ../init.d/quota /etc/rc2.d/S20quota
```

```
root@linux / # ln -s ../init.d/quota /etc/rc3.d/S20quota
root@linux / # ln -s ../init.d/quota /etc/rc4.d/S20quota
root@linux / # ln -s ../init.d/quota /etc/rc5.d/S20quota
root@linux / # ln -s ../init.d/quota /etc/rc6.d/K85quota
```

Zur Festlegung und Verwaltung der Kontingente existiert das Programm `edquota`.

```
root@linux / # /edquota -u user007
```

Dieser Aufruf öffnet den Standardeditor (i.d.R. vi), mit Hilfe dessen man die Änderungen an den Parametern vornehmen kann.

/tmp/EdP.auS8yzc

```
Disk quotas for user user007 (uid 1000):
Filesystem    blocks     soft    hard   inodes   soft    hard
/dev/hda7      80579   100000  120000     3841  10000   12000
```

Detaillierte Informationen findet man in den man-Pages, deshalb seien hier nur in Kurzform die wichtigsten Kommandos aufgezählt:

```
# Ändern der Quota für Benutzergruppe users

root@linux / # edquota -g users

# Ändern der Grace Period (Zeitraum/Frist für Softlimit)

root@linux / # edquota -t

# Übernahme der Quotaeinstellungen von Benutzer user007
# für alle User mit UID > 1000

root@linux / # edquota -p user007 `awk -F: '$3 > 1000 {print $1}'
/etc/passwd`

# Erzeugen eines Reportes

root@linux / # repquota -avug

Report for user quotas on device /dev/hda2
Block grace time: 7days; Inode grace time: 7days

                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   69841       0       0            9226     0     0
postgres  --       1       0       0               1     0     0
ai114     --   80154   60000*  90000  6days     3841* 3840  7000   3days
user007   --   80579  100000  120000            3841 10000 12000


# Statusabfrage für einen Benutzer
```

```
root@linux / # su user007
user@linux / $ quota

Disk quotas for user user007 (uid 1000):
Filesystem  blocks   quota   limit   grace   files   quota   limit
grace
 /dev/hda7   80579   100000  120000          3841    10000   12000
```

Es sei darauf hingewiesen, dass nicht bei allen Dateisystemen Quotas unterstützt werden. Auf den FAQ-Seiten des ReiserFS findet man weitere Informationen.

7.6 Verschlüsselte Dateien

Im Kapitel "GnuPG Handbuch" wurde bereits detailliert auf das Verschlüsseln von Dokumenten eingegangen. Deshalb soll an dieser Stelle nicht näher darauf eingegangen werden. Das folgende Kapitel beschäftigt sich deshalb mit der Einrichtung verschlüsselter Dateisysteme, die an jede beliebige Stelle im Verzeichnissbaum eingehängt werden können. Alle Dateien in einem solchen Verzeichnis werden automatisch ver- und entschlüsselt.

7.7 Verschlüsselte Dateisysteme

Es gibt zwei Möglichkeiten, Dateisysteme zu verschlüsseln:

1. Das Filesystem residiert in einer verschlüsselten Datei
2. Das Filesystem residiert auf einer verschlüsselten Partition

Für beide Varianten sind Modifikationen am Kernel erforderlich, bei Kernel 2.6.XX werden alle passenden Module mitgliefert (Menü "Cryptographic Options"). Für den Fall a) ist zusätzlich der "Loopback Device Support / Cryptoloop Support" im Menü "Block Devices" zu aktivieren. Bei älteren Kerneln ist ein spezieller Patch einzuspielen.

Weiterhin ist es (noch) notwendig, einige wichtige Programme wie `mount` und `losetup` linux-utils zu modifizieren. Keine Probleme gab es mit dem Patch von http://therapy.endorphin.org/patches/ .

```
root@linux / # tar zxvpf util-linux-2.11y.tar.gz
root@linux / # patch -p0 < util-linux-2.11y.losetup.diff
root@linux / # cd util-linux-2.11y
root@linux / # make && make install
```

Die Verschlüsselung wird durch ein virtuelles Laufwerk (Loop-Device) realsiert. Dieses wird, wie jedes andere Laufwerk auch, in den Verzeichnisbaum eingehängt, z.B. an /mnt/crypted. Alle nach `/mnt/crypted` kopierten Dateien werden nun automatisch verschlüsselt. Natürlich muss dieses virtuelle Laufwerk mit einer real existierenden Datei oder Partition verbunden sein, die verschlüsselten Daten sollen ja schließlich nicht verloren gehen.

```
# Fall a) Erzeugen einer Datei (Größe 8MB)

root@linux / # dd if=/dev/zero of=/home/user/crypedFile.dat bs=4096
count=2048
```

```
# Erzeugen eines leeren Verzeichnisses, welches
# sensible Daten aufnehmen soll
root@linux / # mkdir /mnt/crypted
# mit AES verschlüsseltes loop-Device mit dieser Datei verbinden
root@linux / # losetup -e aes /dev/loop0 /home/user/crypedFile.dat

# oder Partition mit loop-Device verbinden
root@linux / # losetup -e aes /dev/loop0 /dev/hda7
# auf dem virtuellen Laufwerk ein Dateisystem anlegen
root@linux / # mkfs -t ext2 /dev/loop0
# virtuelles Laufwerk mounten
root@linux / # mount /dev/loop0 /mnt/crypted
# nun kann normal gearbeitet werden:
root@linux / # cp datei /mnt/crypted
# am Ende: unmount
root@linux / # umount /dev/crypted
# virtuelles Laufwerk (loop-Device) trennen
root@linux / # losetup -d /dev/loop0
```

Um das Einhängen der Laufwerke zu automatisieren, kann die Datei /etc/fstab angepasst werden (letzte Zeile).

/etc/fstab

```
# <Dateisystem>           <Mountpunkt>  <Typ>  <Optionen>                  <dump>  <pass>
/dev/hda2                 /             ext3    defaults,errors=remount-ro    0       1
proc                      /proc         proc    defaults                      0       0
/dev/fd0                  /floppy       auto    defaults,user,noauto          0       0
/dev/cdrom                /cdrom        iso9660 defaults,ro,user,noauto       0       0
/home/user/crypedFile.dat /mnt/crypted  ext2    defaults,loop,encryption=aes  0       0
```

Natürlich gibt es auch Alternativen zu der hier vorgestellten Vorgehensweise, beispielsweise findet man auf pro-linux eine Anleitung zu Loop-AES.

Für die Zukunft bleibt zu hoffen, dass die Einrichtung von verschlüsselten Dateisystemen sich um Größenordnungen vereinfacht.

8 Die Firewall

8.1 Was ist eine Firewall?

Lutz Donnerhacke schreibt in der FAQ zur Newsgroup de.comp.security.firewall

> Als Firewall bezeichnet man ein organisatorisches und technisches Konzept zur Trennung von Netzbereichen, dessen korrekte Umsetzung und dauerhafte Pflege. Ein oft benutztes Instrument der Umsetzung ist ein Stück Hardware, das zwei physisch getrennte Netzbereiche genau so verbindet, wie es im Konzept zugelassen wird. Dieses Stück Hardware bezeichnet man als Firewall-Rechner/System oder verkürzt als Firewall.

8.2 Paketfilter und Kernelparameter

Der Paketfilter ist ein Teilaspekt der Firewall, der Sicherheitsrichtlinien bezüglich des zu transportierenden Netzwerkverkehrs durchzusetzen versucht und idealerweise die einzige Verbindung zu einem nicht vertrauenswürdigen Netz darstellt. Dies beinhaltet natürlich auch den Netzwerkverkehr vom und ins Internet. Anhand festgelegter Regeln werden Datenpakete (Datagramme) gefiltert, protokolliert (logging), markiert und geändert. Bei der Formulierung dieser Regeln für das Netzwerk gelten beispielsweise folgende Überlegungen:

* wer nutzt das Netzwerk?
* welche Daten dürfen welche Wege nehmen?
* welche Dienste werden von wem und für wen angeboten?
* wann werden die Dienste angeboten?
* ...

Bei der Beantwortung dieser Fragen werden auch die Maßnahmen klar, die zu treffen sind, um das Netzwerk korrekt zu konfigurieren. Es werden IP-Adressen vergeben, Routen gesetzt, Dienste in der Form konfiguriert, dass sie auch nur dort angeboten werden, wo sie genutzt werden sollen (Freigaben zum Beispiel nur im LAN und nicht im Internet). Erst wenn diese grundlegende Netzwerkkonfiguration getätigt ist, besteht die Möglichkeit, dass man die Hilfe des Paketfilters benötigt, um den Verkehr im Netzwerk zu steuern. Man sollte sich jedoch bewusst sein, dass Paketfilter auf Schicht 3 und 4 des ISO/OSI Modells arbeiten. Um Entscheidungen aufgrund von Informationen der Applikationsebene (Schicht 7) wie z.B. `http` oder `irc` treffen zu können, werden Proxies eingesetzt, die das Thema des nächsten Abschnittes sind.

8.3 Proxy

Paketfilter sind aber bei weitem nicht die einzige Maßnahme, die zur Regelung des Netzwerkverkehrs zur Verfügung steht. Weitere Möglichkeiten bietet der Proxy (Application Level Gateway), der sich zwischen Client und Server befindet. Er arbeitet auf Applikationsebene (Schichten 5-7 des ISO/OSI Referenzmodells) und kennt die Interna der jeweiligen Protokolle (z.B. `http`, `irc`). Was bedeutet das? Der Client kommuniziert nicht direkt mit dem Server, sondern er verbindet sich zu dem Proxy und weist diesen an, eine Verbindung mit dem Server aufzunehmen. Der Proxy sendet und empfängt Daten vom Server, filtert diese nach bestimmten Regeln und leitet sie dann an den Client weiter, oder auch nicht. Proxies realisieren eine logische Trennung der Kommunikationspartner, es existieren also zwei unabhängige Verbindungen: Client-Proxy und Proxy-Server. Dadurch werden

* Authentifizierung und Autorisierung (benutzerabhängige Nutzung von Diensten)
* Zwischenspeicherung (Cache) von Daten
* Filterung von Dateninhalten (Virenscanner, Kindersicherung)

* Löschen der Datenherkunft (Anlegen von personenbezogenen Profildaten unterbinden)

ermöglicht. Normalerweise muss der Proxy vom Client unterstützt werden, aufgrund des Design einiger Netzwerkprotokolle lassen sich diese nicht oder nur schwer über einen Proxy leiten. Es gibt aber auch generische Proxy-Protokolle, wie zum Beispiel SOCKS5. Eine interessante Lösung stellt ein transparenter Proxy dar: Die Client-Server Verbindung wird durch den Paketfilter über den Proxy umgeleitet, ohne dass der Client davon etwas merkt.

Als repräsentative Vertreter seien an dieser Stelle

* SOCKS (Dante) als generischer Proxy
* squid und wwwoffle als HTTP-Proxies,
* Exim und pop3gwd als Email-Proxies.
* bnc und tircproxy als IRC-Proxies

genannt.

8.4 Was können Firewalls und was können sie nicht?

Eine Firewall kann unter anderem...

* den Netzwerkverkehr zwischen Netzen kontrollieren und einschränken,
* Netzwerkzugriffe auf Rechner bzw. Dienste zulassen oder blockieren,
* den Netzwerkverkehr protokollieren,
* den Netzwerkverkehr manipulieren (z.B. Umleitung auf andere Rechner bzw. Proxies).

Ein Firewall kann nicht dazu beitragen,

* alle Sicherheitslücken im System / Netzwerk zu schließen,
* Konfigurations- oder Installationsfehler zu beseitigen,
* Systemanomalien (Viren, Trojaner, Würmer,...) oder Einbrüche festzustellen und
* Schwachstellen (z.B. einfache Passwörter) zu erkennen.

Keine Firewall zu haben ist in jedem Falle ehrlicher, als eine Firewall aufzustellen, und damit zu glauben, man sei sicher!

9 Testen der Sicherheitsmaßnahmen

Für das Testen der Sicherheitsmaßnahmen können aufgrund des sehr differenzierten Sicherheitsbedürfnisses und der sehr unterschiedlichen Ausrichtung der Systeme nur bedingt allgemeingültige Richtlinien festgelegt werden. Deshalb werden hier nur einige wenige Maßnahmen aufgeführt, die weder für alle Systeme gelten noch für alle Systeme ausreichend sind. Sie sollen lediglich als Einstieg einer kontinuierlichen Prüfung angesehen werden.

Der Systemtest erfolgt durch "Experimentelle Prozessanalyse", also durch (simulierte) Angriffe gegen den eigenen Rechner oder das eigene Netzwerk, die natürlich außerhalb des eigentlichen Systembetriebes stattfinden sollten. Damit können Einbruchserkennungssysteme (IDS), Firewalls (Paketfilter) und die Systemstabilität getestet werden. Allerdings werden auf diesem Weg niemals alle Schwachstellen gefunden, weil diese entweder noch nicht bekannt sind (z.B. Buffer Overflows, unbekannte Viren) oder weil ein Angriff nicht zu verantworten ist (Elektrische Überspannung zerstört unter Umständen Hardware).

Bei Netzwerksystemen sollte der erste Angriff ein Portscan (z.B. mit dem Tool `nmap`) sein. Anschließend ist eine Analyse des Datenverkehrs (Sniffer) denkbar, zumindest wenn vertrauliche Informationen über das Netzwerk übertragen werden. Eine Vielzahl an Angriffsszenarien bietet der Security Scanner Nessus. Die Schwachstellenanalyse erfolgt durch derzeit 1049 Plugins aus 23 Kategorien (Ende 2002), die natürlich auch andere Systeme wie Windows-Rechner oder Cisco Router untersuchen. Zu den interessantesten Plugins gehören:

Backdoors

* `mstream`, `shaft`, `TFN`, `Trin00`, `trinityV3` (alles DDOS-Tools)
* alya.cgi (verwendet von vielen root-Kits)

Denial of Service

* Teardrop
* Bonk
* BlackIce DoS (ping flood)
* Generic flood

Firewalls

* icmp Angriffe
* Proxy Angriffe entfernter Shell-Zugriff
* Apache-SSL, Squid, SSH Buffer Overflows
* MySQL Schwachstellen entfernter Superuser-Zugriff
* Samba: Entfernte Erstellung beliebiger Dateien
* thttpd 2.04 buffer overflow
* HTTP header overflow

misc

* X Server
* Apache Serverstatus verfügbar
* RedHat 6.2 inetd

CGI Missbrauch

* Apache Tomcat /servlet ...
* Oracle 9iAS ...

* phpMyAdmin Zugriff ...

Die Angriffe sollten selbstverständlich abgewährt und gegebenenfalls protokolliert werden. Eine detailliertere Beschreibung dieses Security-Scanners wird zu einem späteren Zeitpunkt in einem eigenständigem Kapitel erfolgen.

10 Systemüberwachung (Logging und Accounting)

10.1 Einleitung Systemüberwachung

Eine Grundforderung an vertrauenswürdige Systeme ist die Nachvollziehbarkeit des Systemverhalten. Bei Linux werden verhaltensrelevante Ereignisse in so genannten Logfiles gespeichert, die meist unter /var/log zu finden sind. Die wichtigsten Systemlogfiles sind:

* messages
* auth.log
* kern.log

In den Protokolleinträgen finden sich Hinweise, warum sich z.B. ein Dienst nicht starten läßt oder das mehrere Login-Versuche fehlgeschlagen sind. Meist erfolgt zu jedem Protokolleintrag eine Zeitangabe.

So wie der `xinetd` als Superserver für das Starten der Dienste zuständig ist, gibt es auch einen Protokolldienst, den `syslogd`. Dieser wurde äußerst detailliert in einem eigenständigen Selflinux- Kapitel beschrieben und soll deshalb an dieser Stelle nicht nochmals betrachtet werden. Thema dieses Abschnittes sind Mechanismen zum Schutz der Logfiles. (Kernelmeldungen werden vom `klogd` entgegengenommen und verarbeitet, das Programm `dmesg` gibt diese Meldungen aus.)

Weitaus mehr Schutzmöglichkeiten bietet die neue Variante von `syslog`, der syslog-ng. Dieser ist jedoch derart komplex, dass sich in Zukunft ein eigenständiges Kapitel damit beschäftigen wird.

Sehr interessant auch die Kombination von `PHP`, `mySQL` und `syslog-ng`, mit welcher sich das Open Source Projekt php-syslog-ng beschäftigt.

10.2 Schutz vor "Überlaufen" der Logfiles

Eine Angriffsmöglichkeit gegen Protokollierungsmechanismen besteht darin, wahnsinnig viele Log-Einträge zu generieren, so das

1. der begrenzte Speicherplatz aufgebraucht und
2. die Logfiles sehr unübersichtlich

werden. Abhilfe verspricht das Programm `logrotate`, welches Log-Dateien rotieren lässt. Was heißt das? Wenn ein Logfile eine bestimmte Größe oder ein bestimmtes Alter erreicht hat, wird es komprimiert, gesichert und umbenannt. Die nun zu protokollierenden Meldungen werden in einem neu angelegten (leeren) Logfile gesichert. Die Anzahl der Rotationsvorgänge, nach denen ein Logfile-Backup endgültig gelöscht wird, kann man einstellen. In Kombination mit dem Schutz vor DOS-Attacken des `xinetd` (instances, per_source) kann ein Überlaufen der Logfiles verhindert werden, und die sicherheitsrelevanten Einträge bleiben bei einem versuchten Einbruch erhalten.

10.3 Schutz vor Löschen und Manipulation der Einträge

Drucker

> Falls ein alter Drucker zur Verfügung steht, kann man diesen für die Protokollierung besonders sensitiver Logfiles benutzen. Die Konfiguration erfolgt innerhalb der /etc/syslogd.conf beziehungsweise der /etc/syslog-ng.conf .

Loghost

Logfiles können auf einem dedizierten zentralen Server abgelegt werden. Einen solchen Rechner nennt man Loghost. Damit dieser Server auch Meldungen anderer Rechner entgegen nimmt, muss man den `syslogd` mit dem Parameter `-r` starten. Die Kommunikation zwischen Client und Loghost erfolgt auf Port 514/udp, und genau dies eröffnet einem Angreifer große Möglichkeiten (Beobachtung, Fälschen von Meldungen, DOS-Attacke). Deshalb sollte der Loghost niemals von außen erreichbar sein. Detaillierte Informationen zur Konfiguration und zu den damit verbundenen Gefahren finden Sie im syslog Kapitel.

Nun, welche Sicherheitsmaßnahmen sollten unbedingt erfolgen:

* Der Loghost sollte nur per Konsole zugänglich sein (nicht `ssh`, `http`, `ftp` oder Ähnliches). Somit kann ein Angreifer von Außen keine Log-Einträge löschen.
* Der Loghost ist der Loghost und nichts als der Loghost. (also keine weiteren Dienste wie `Routing`, `Mail` oder `DNS`)
* Der Loghost sollte zusätzlich durch einen für seine Anforderungen angepassten Paketfilter abgeschottet sein.
* Der Loghost sollte "nicht sichtbar sein" und somit nicht auf PING-Anfragen (ICMP Messages) antworten. (`echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all`)

Folgende Maßnahmen erfordern relativ großen Aufwand:

* Erlangt ein Angreifer die Macht über einen Client, so findet er in dessen sylog.conf den Hinweis auf den Server. Um diese Schwachstelle zu beseitigen, könnte man den `syslog` Dienst auf jedem Client derart manipulieren, dass dieser eine andere Konfigurationsdatei lädt. Dazu muss man natürlich den syslog-Quellcode patchen und neu übersetzen.... Darauf wird in einem später folgenden Artikel detaillierter eingegangen.
* Die Kommunikation zwischen Loghost und angeschlossenen Clients sollte verschlüsselt werden (z.B. `stunnel`).

Eine englischsprachige Kurzanleitung unter http://www.campin.net/newlogcheck.html beschreibt den Aufbau eines Loghosts mit `syslog-ng`, `mysql`, `swatch` und `stunnel`.

10.4 Überwachung der Verbindungen (Login / Connection Accounting)

Die Verbindungszeiten der Systembenutzer, also WANN WER WO eingeloggt war, werden in der Datei /var/log/wtmp gespeichert. Da diese jedoch binär vorliegt, benötigt man das Programm `last`.

```
user@linux / $ last

root   tty1     Sun Sep    3   11:45 -12:14 (00:29)
ai114  tty7     Sun Sep    3   13:45 -15:56 (02:11)

wtmp begins Tue Jan 15 13:54:09 2003
```

Mit `lastlog` wird die Datei /var/log/lastlog in menschenlesbarer Form ausgegeben.

```
user@linux / $ lastlog

Benutzer    Port   Von              Letzter
root               tty1             Mon Jul 14 21:05:29 +0200 2003
daemon                              **Nie angemeldet **
```

```
bin                                  **Nie angemeldet **
ai114        :0                      Die Jul 15 12:58:19 +0200 2003
mysql                                **Nie angemeldet **
```

10.5 Überwachung der Prozesse (Process Accounting)

Im Gegensatz zur Verbindungsüberwachung lassen sich beim Process Accounting genaue Aussagen über genutzte Systemressourcen (CPU, Speicher, IO) treffen.

Die Programme aus dem Paket `acct` erlauben detaillierte Aufstellungen der verbrauchten Ressourcen sowie einer Zuordnung zu einem Systembenutzer und natürlich dem genauen Zeitpunkt (minutengenau).

Connection Accounting muss explizit im Kernel aktiviert werden (General Setup --> BSD Process Accounting). Die eigentlichen Aufzeichnungen übernimmt jedoch wieder ein Daemon, er wird durch den Aufruf

```
root@linux / # accton /var/log/logfile
```

gestartet und kann durch den Aufruf ohne zusätzliche Optionen wieder deaktiviert werden. (Optional SysV-Initskript analog zu "`quota`")

```
# Start der Aufzeichnung

root@linux / # accton /var/log/psacct

# Ende der Aufzeichnung
root@linux / # accton
```

Für die Auswertung stehen drei Programme zur Verfügung:

lastcomm	vollständige Aufzeichnung aller Prozesse (user, tty, CPU Zeit, Zeitpunkt)
sa	statistische Zusammenfassung (verschiedene Sortieroptionen)
ac	Statistik der Verbindungszeiten

```
#Ausgabe aller Prozesse für Benutzer user007

root@linux / # lastcomm user007

Eterm                  user007    ??         2.05 secs Wed Jul 16 17:34
bash                   user007    ??         0.43 secs Wed Jul 16 17:34
bash                F  user007    ??         0.00 secs Wed Jul 16 17:56
tty                    user007    ??         0.03 secs Wed Jul 16 17:56
mozilla-bin         F  user007    ??         0.00 secs Wed Jul 16 17:52
mozilla-bin         F  user007    ??         0.00 secs Wed Jul 16 17:52
mozilla-bin         F  user007    ??         0.00 secs Wed Jul 16 17:52
mozilla-bin         F  user007    ??         0.02 secs Wed Jul 16 17:52
quota                  user007    ??         0.04 secs Wed Jul 16 17:43
rm                     user007    ??         0.15 secs Wed Jul 16 17:43
```

```
# Statistik nach Nutzern

root@linux / # sa -m

ai114                 2644 14363398.28re    2248.20cp         0avio
10689k
root                 22701  6829449.23re     314.14cp         0avio
552k
mail                    54      984.15re       0.14cp         0avio
1006k
user007                 15        3.78re       0.02cp         0avio
505k
mysql                   20        0.35re       0.00cp         0avio
17392k


# Statistik nach IO-Operationen sortiert:

root@linux / # sa -ad

25450 21211810.82re    2563.42cp         0avio      1621k
  3508    4565.43re       8.46cp         0avio       340k   grep
  2183   23553.40re       1.74cp         0avio       347k   mgetty
  1999     272.72re       0.76cp         0avio       331k   mv
  1973     320.78re       0.77cp         0avio       297k   basename
  1957  162216.27re       6.55cp         0avio       644k   sh
  1483 4579495.53re       6.76cp         0avio     18518k   mozilla-bin*
   923     255.08re       0.47cp         0avio       363k   rm
   793   19297.68re       0.57cp         0avio       373k   gcc
   ...
   ...


# Verbindungszeiten

root@linux / # ac -d

Jul 15  total        9.29
Jul 16  total       14.55
Jul 16  total        1.20
Jul 16  total        0.91
Today   total        7.86


#Verbindungszeiten aller Benutzer:

root@linux / # ac -p

root                            1.47
user007                        32.37
total       33.84
```

Weitere Informationen sind wie gewohnt in den man-Pages zu finden (ac(1), accton(8), lastcomm(1) und sa(8)).

11 Einbruchserkennung (Intrusion Detection)

11.1 Einführung Einbruchserkennung

Nachdem die bisher behandelten Schutzmaßnahmen getroffen wurden, gilt es nun, deren Effizienz kontinuierlich zu verbessern und Schwachstellen, Angriffsversuche oder gar erfolgreiche Einbrüche frühzeitig zu erkennen. Dabei helfen Einbruchserkennungssysteme (IDS), die entweder einzelne Rechner (Hostbasierte IDS) oder den Netzverkehr (Netzwerkbasierte IDS) überwachen. Das Ziel von IDS-Systemen besteht also im Aufdecken von unerlaubten Aktionen gegen ein Netzwerk oder einen Rechner. Normalerweise sind dazu die folgenden drei Schritte notwendig:

* Datensammlung
* Datenanalyse
* Ergebnisdarstellung

11.2 Hostbasierte IDS (HIDS)

HIDS überwachen Daten und Prozesse auf einem einzelnen Rechner. Sie beobachten Logfiles, offene Netzwerkschnittstellen und Netzwerkverbindungen des Computers.

Eine Möglichkeit der Einbruchserkennung stellen die sogenannten System Integrity Verifier (SIV) dar, welche Dateien und Verzeichnisse anhand von Hashwerten auf deren Echtheit und Originalität prüfen. Damit können Trojaner, Viren und Root-Kits gefunden werden. Eine ganz einfachen SIV kann man mit dem `ls`-Kommando folgendermaßen selbst entwickeln:

Das Ausführen von

```
root@linux / # ls -ailR /etc >/root/etc.orginal
```

erzeugt eine Datei, welche neben Dateigröße und Dateirechten auch die Nummer der Inode enthält. Diese Datei wird auf einem Medium mit Schreibschutzmöglichkeit (Diskette, CD-ROM) abgelegt.

Über Veränderungen informiert das folgende Kommando:

```
root@linux / # ls -ailR /etc | diff /root/etc.original
```

wobei hier natürlich der Ablageort der Datei angepasst werden muss. Wichtig sind neben dem /etc Verzeichnis auch die mit dem t-Bit (Sticky-Attribut) versehen Dateien, die mit

```
root@linux / # find / -type f -perm -6000 -exec /bin/ls -ail{} \;
>/root/original
```

gesichert werden. Die Auswertung der Log-Files gehört ebenfalls zu den Maßnahmen, die man ohne Weiteres realisieren kann. Zur Übertragung dieser Protokollinformationen sollte man den sichereren `ssyslog` in Betracht ziehen. Da ein Angreifer natürlich auch eine Datei modifizieren kann ohne die Dateigröße zu ändern, sollten bessere Mechanismen wie beispielsweise Hashwerte zum Einsatz kommen. Diesen Ansatz verfolgen beispielsweise die `md5-tools` und `debsums` sowie das mittlerweile auch in einer freien Version (GPL) verfügbare `Tripwire`. Letztgenanntes bietet sehr feingranulare Zugriffsmechanismen. So kann man mit mehr

als 15 Dateiattributen detaillierte Richtlinien festlegen, z.B. schützt man mit
```
/sbin/iptables -> $(SEC_KRIT);
/tmp -> $(SEC_INVARIANT)
```
den Paketfilter iptables vor jeglicher Manipulation und das temporäre Verzeichnis /tmp vor Änderung des Besitzers und der Zugriffsrechte. Solche Regeln sind bereits vordefiniert und müssen nur noch angepasst werden. Eine sehr gute Dokumentation findet man auf den Internet-Seiten des Linux Magazins.

Weiterhin sei an dieser Stelle auf LIDS (Linux Intrusion Detection System) verwiesen. LIDS versucht die Schwächen von Linux insbesondere die Allmacht des Superusers "root" zu beseitigen. Ein Kernel-Patch erweitert den Kernel um differenzierte Sicherheitseinstellungen, sodass Dateien, Prozesse und Teile der Systemverwaltung (Firewall-Regeln, Routing) besser geschützt werden können. Weitere Zugriffssteuerungsmodelle wurden in SELinux (Security Enhanced Linux) umgesetzt.

11.3 Netzwerkbasierte IDS (NIDS)

NIDS erkennen Angriffe im Netzwerk indem sie den Netzwerkverkehr analysieren. Damit sind diese nicht nur auf den Rechner beschränkt, auf dem sie installiert wurden.

Einen Überblick über das schützende Netzwerk erlangt man durch den Einsatz von Netzwerkmonitoren und Paketsniffern, zu denen ntop und ethereal gehören. Als Vertreter für ein netzwerkbasiertes Einbruchserkennungssystem soll im Folgenden das freie snort kurz vorgestellt werden.

`Snort` sucht im Netzwerk nach bekannten Angriffsmerkmalen und -mustern. Diese Merkmale werden in Form von paketbeschreibenden Regellisten (Signaturen) angegeben und können von der `snort` Webseite heruntergeladen werden. Mit Modulen, den sogenannten Präprozessoren, kann man `snort` jederzeit erweitern. Eines dieser Module ist `frag2`, das Paketfragmente zusammensetzt und somit die Grundlage einer erfolgreichen Mustererkennung für zerstückelte Angriffsmuster bietet. Ein weiteres wichtiges Modul ist `portscan`, durch das Ausspähungsversuche sichtbar werden. Der Präprozessor `spade` (Statistical Packet Anomaly Detection Engine) ist ein selbstlernendes Modul, das nach einer Lernphase anormalen Netzwerkverkehr feststellen kann.

Ein entscheidendes Kriterium für den Erfolg eines NIDS ist die Platzierung des Sensors (z.B. des `snort` Systems). Dazu sollte man sich erst einmal ein Bild von der Topologie des Netzes machen. (Stern, Bus, Ring, ...) und die Stellen im Netz markieren, die überwacht werden sollen. Bei dem Einsatz eines Switch (Stern) ergeben sich unweigerlich Probleme bei der Überwachung, da der Verkehr nicht zwangsläufig über den Sensor geleitet wird. Abhilfe schaffen hier Monitor-Ports, die allerdings nur an teuren Geräten zu finden sind. Eine billigere Variante wäre ein HUB, an dem die zu überwachenden Rechner und das NIDS-System (`snort`-Rechner) angeschlossen werden.

11.4 identd

Der `identd` Server beobachtet TCP-Verbindungen und ermittelt die Benutzerkennung des Prozesses, der die Verbindung benutzt. (Nähere Informationen RFC 1.1)

Mit diesen Informationen lassen sich Angriffe besser nachvollziehen und ausgenutzte Sicherheitslöcher besser lokalisieren. Allerdings dienen solche Informationen auch einem Angreifer, der damit z.B. die UID des Webservers abfragt. Läuft dieser Server mit root-Rechten, so ist er ein "lohnendes Ziel". Der Einsatz des `identd` sollte also immer genauestens abgewogen werden.

12 Notfallplan im Falle einer Systemkompromittierung

12.1 Grundsätzliches

Wenn man einen erfolgreichen Einbruch entdeckt, ist es äußerst wichtig, die Spuren zu sichern, damit man den Tathergang nachvollziehen kann. Dies ist die Voraussetzung, um wirksame Gegenmaßnahmen bestimmen und durchführen zu können. Mit dieser Thematik beschäftigt sich die "Digitale Forensik", die an dieser Stelle nicht detailliert vorgestellt werden kann. Wir versuchen in dieser kurzen Einleitung lediglich, einige wichtige Grundaussagen mit praktischen Tipps zu verbinden.

Nach der Entdeckung eines Einbruchs ist es besonders wichtig, Ruhe zu bewahren und möglichst keinerlei hastige Eingriffe zu tätigen, da bereits beim Eingeben von Kommandos wichtige Informationen zur Rekonstruktion des Angriffes verloren gehen können.

12.2 Protokollierung des Einbruches und der Gegenmaßnahmen

Es sollten einige grundsätzliche Informationen unbedingt schriftlich festgehalten werden, damit man später auch vor Gericht Beweise vorlegen kann:

* Wann wurde der Einbruch festgestellt (Datum, Uhrzeit)?
* Welche Systeme sind / waren betroffen?
* Wer bemerkte den Einbruch?
* Welche Maßnahmen wurden durch wen eingeleitet?
* .....

Die Protokollierung sollte für jeden einzelnen Schritt der Angriffsrekonstruktion mit genauer Angabe von Datum und Uhrzeit beibehalten werden.

12.3 Isolation des Rechners / Subnetzes

Bei besonders sensiblen Daten ist es unter Umständen das Wichtigste, das kompromittierte System vom Netz zu trennen, indem man einfach das Netzwerkkabel entfernt. Auf das Herunterfahren des Rechners sollte unbedingt verzichtet werden, da ansonsten vom Angreifer manuell gestartete Prozesse (oder eingebundene Kernelmodule) nach einem Neustart nicht mehr aktiv sind.

12.4 Backup und Rekonstruktion

Die Sicherung eines kompromittierten Systems ist die Grundlage für das Nachvollziehen eines Angriffes. Äußerst wichtig ist es, dass alle Textein- und Ausgaben aufgezeichnet werden. Graphische Ausgaben sollten per Screenshots gesichert werden (z.B. mit `xwd` oder `gimp`).

Man sollte einen aktuellen Schnappschuss (Snapshot) des Systems auf einem Backup-Medium erzeugen. Die Sicherung der Daten sollte in Reihenfolge ihrer Vergänglichkeit erfolgen:

* Cache, Hauptspeicher
* Netzwerkverbindungen
* Prozesse
* Festplattendaten
* Daten auf Disketten, CD-RW, Streamer, ...

* Daten auf CD-R, Notizen, Post-It-Notes, ...

Die "Spurensicherung" und Analyse wird durch verschiedene Softwaresammlungen unterstützt. Zu diesen gehören unter anderem TCT, TCTUtils, und cryptcat. Auf keinen Fall sollten die möglicherweise kompromittierten Programme des Opferrechners zum Einsatz kommen, viele Root-Kits tauschen Systemprogramme wie ps und `lsmod` einfach aus. Die sichere Alternative besteht darin, eine "saubere" Rettungsdistribution von Diskette oder CD-Rom zu mounten und deren Systemprogramme zu verwenden. (Nicht Booten sondern nur Mounten!!!) Zu empfehlen sind hier neben Knoppix auch sogenannte "Business Cards", also Rettungs-CDs in Scheckkartengröße. Zwei Vertreter können als ISO-Image unter http://www.lnx-bbc.org/ bzw. http://www.inside-security.de/INSERT.html heruntergeladen werden.

Folgende Maßnahmen sind bei den meisten Einbrüchen sinnvoll:

Aufzeichnung aller Sicherungsaktionen auf einen anderen Rechner
```
mkfifo named_pipe
script -f named_pipe
cat named_pipe | nc LogHost LogPort
```

Datensammlung auf anderen Rechner mittels `nc` (netcat)

* kompromittierter Rechner:
  `[Daten] | nc -w 2 [LogHost-IP] 6666`
* LogHost:
  `nc -l -p 6666 | gzip >> datei.gz`

Bei nicht vertrauenswürdigen Netzen sollte die Kommunikation verschlüsselt werden:

* kompromittierter Rechner:
  `[Daten] | des -e -c -k [Schlüssel] | nc -w 2 [LogHost-IP] 6666`
* LogHost:
  `nc -l -p 6666 | des -d -c -k [Schlüssel] >> Datei`

z.B. Hauptspeicher:

* kompromittierter Rechner:
  `dd if=/dev/mem | nc -w 2 192.168.1.5 6666`
* LogHost:
  `nc -l -p 6666 > mem.img`
* Prozesse:
  `ps enf -Aelf --cols 1000 ls -lR /proc/[0-9]*`

Konfiguration der Netzwerkkarten
```
ifconfig -a
netstat -iea
ip addr show
```
(iproute2-Utilities)

Routingtabellen
```
netstat -rn
iproute2-Utilities
```

* ip route show table main
* ip rule show (iproute2-Utilities)

Paketfilterregeln
```
iptables -L -vn --line-numbers
```

geöffnete Dateien
```
lsof
```
aktuelle Netzwerkverbindungen
```
lsof -i
```
Sockets
```
lsof -U
```
Gelöschte aber noch offene Dateien
```
lsof +L1
```

Einbrecher können ein Programm auch nach dessen Start löschen. Man kann bei installiertem proc- Filesystem gerade ablaufende Programme wieder in eine Datei sichern:
```
cat /proc/[PID]/exe > programm.bin
```

weiterhin:

* `last` (letzter eingeloggter Benutzer)
* `who` (aktuelle Benutzer)
* `w` (welcher Benutzer führt was aus?)
* `arp`
* `fdisk`
* .....

Partition übers Netz sichern:

* kompromittierter Rechner: `dd if=/dev/hda1 | nc -w 2 [LogHost-IP] 6666`
* LogHost: `nc -l -p 6666 | gzip >> hda1.img.gz`

12.5 Analyse des Angriffs

Nun hat man alle wichtigen Informationen gesichert und kann selbst versuchen, den Einbruch nachzuvollziehen. Dadurch schafft man die Grundlage, um einem erneuten Eindringen vorzubeugen. Alternativ dazu kann man mit speziellen Überwachungsfunktionen einen wiederholten Einbruch automatisch und damit sehr schnell aufdecken. (honeypot / Honigtopf).

Folgende Informationen müssen nun ausgewertet werden:

* veränderte Konfigurationsdateien
* veränderte Systemprogramme
* eingeschleuste Programme (meistens in versteckten Ordnern)
* Sniffer
* Logfiles
* Speicherinhalt / offene Netzwerkverbindungen

In die Auswertung müssen natürlich alle vom Opfersystem erreichbaren Rechner einbezogen werden. Dies beinhaltet sowohl Netzwerkrechner als auch Systeme mit einem Remote-Zugang (z.B. Laptops).

Zu den konkreten Maßnahmen, die durchzuführen sind, gehören unter anderem:

* Auflisten aller Dateien / Verzeichnisse mit s-Bit (Finden von root-Shells)
```
find / -type f \(perm -04000 -o -perm -02000\) \exec ls -lg {}\;
```

* Herausfinden von Dateien / Verzeichnissen mit ungewöhnlichen Namen
  `find / -name ".." -print find / -name "..." -print find / -name ".mail" -print`
* Herausfinden von Remote-Zugriffen des Superusers
  `grep "uid=0" /var/log/*`
* Herausfinden aller ssh-Verbindungen von unbekannten Rechnern (Prinzipiell sollte der Remote Zugriff nur von bekannten Rechnern ausgehen)
  `grep "Connection" /var/log/*`
* Herausfinden einer gezielten Manipulation der Logfiles (abruptes Enden von Logfiles deutet auf Manipulation)
  `head /var/log/messages head /var/log/secure`
* Kontrolle von `passwd`, `hosts.allow`, hosts.deny, inetd.conf, xinetd.conf auf mögliche Veränderungen (Systembenutzer wie z.B. `www-run`, `news`, `sync` dürfen keine Login-Shell besitzen)
* Feststellen eines Sniffers (Netzwerkschnittstelle im Promiscuous Modus) `ifconfig`
  eth0 Linkverkapselung: Ethernet Hwaddr 08:FF:TF:i6:88:51:B9
  inet addr 192.168.1.112 Bcast:192.168.1.255 Mask 255.255.255.0
  UP BROADCAST RUNNING **PROMISC**
  Multicast MTU:1500 Metric:1
  Empfangene ...
* Überprüfung von `cron` und `at` Jobs
* Verdächtige Binaries mit "strings" ansehen
* Manche Angreifer sind ziemlich "blöd"
  `cat .bash_history`

Beim Überprüfen dieser und anderer Einbruchsspuren hilft auch das TCT (The Coroner's Toolkit), welches skriptbasiert sämtliche Aufzeichnungen sammelt. Die gesammelten Informationen muss man allerdings noch manuell auswerten. Eine weitere interessante englischsprachige Quelle ist: http://www.securityfocus.com/infocus/1503

12.6 Meldung des Angriffs

Wen sollte man informieren?

* den Systemadministrator, von dessen Rechner der Angriff ausging (Email an root@ip-Adresse oder root@systemdomain.de)
* alle möglicherweise betroffenen Systemadministratoren
* möglicherweise den ISP (z.B. AOL / T-Online)
* möglicherweise den CERT

Was soll wie gemeldet werden?

Auf den Internetseiten des CERT findet man einen englischsprachigen Vordruck, deshalb sollen hier nur die wichtigsten Angaben aufgeführt werden:

* eigene Email-Adresse
* Firmenanschrift, Tel. und Fax
* eigene IP-Adresse, Hostname und Domain
* Liste aller am Angriff beteiligten Rechner (IP-Adressen)
* genauer Zeitpunkt der Attacke
* detaillierte Beschreibung des Angriffs
* Angaben über Angriffserkennung
* wichtige Logfile-Einträge

* Auflistung der Forderungen (z.B. Bitte, die Angriffe zu unterlassen oder Erklärung, wie es dazu kommen konnte)

13 Fazit

Der Umfang dieses Kapitels zeigt die Komplexität des Themas sehr deutlich. Das Ziel des Kapitels bestand darin, einen groben aber möglichst umfassenden Überblick zur Sicherheit von Linux-Systemen zu vermitteln. Allerdings konnten einige sicherheitsrelevante Bereiche (noch)nicht behandelt werden. Dazu zählen:

* IPSEC und IPv6 - Sicherheit auf IP-Ebene
* Tunneling-Protokolle PPTP sowie L2TP
* ssh
* chroot Umgebung
* BIOS-Sicherheit
* Kerberos
* NIS-Sicherheit
* NFS-Sicherheit
* VPN - Virtuelle Private Netze
* LIDS - Linux Intrusion Detection System
* RAS-Sicherheit (RADIUS/TACACS)
* Wireless Security (IEEE 802.11x)

Diese Liste unterstreicht die Notwendigkeit, dieses Kapitel in Zukunft zu erweitern und zu pflegen, angedacht sind meinerseits halbjährliche Aktualisierungen.