



11 Linux-Intrusion-Detection-System – LIDS

11.1 Einführung

Das Linux-Intrusion-Detection-System (LIDS) ist kein Intrusion-Detection-System im herkömmlichen Sinne. Es verfolgt eine ganz andere Zielsetzung als die klassischen host- oder netzwerkbasierten Systeme.

Der Benutzer *root* ist auf fast allen UNIX-Systemen ein Administrator mit unbeschränkten Rechten. Scherzhaft kann man ihn als den Gott des Systems bezeichnen. Es gibt nur einen (mit der Benutzernummer (UID) 0) und er darf alles. Ein Einbrecher wird bei einem Angriff immer versuchen, *root*-Privilegien zu erhalten. Dies erfolgt entweder bereits beim eigentlichen Angriff über das Netzwerk oder anschließend, nach lokaler Anmeldung als normaler Benutzer, durch die Ausnutzung lokaler Sicherheitslücken. Sind dem Einbrecher vom System die Rechte von *root* übertragen worden, kann er das System nach seinem Gutdünken verändern, Trojanische Pferde und *Rootkits* installieren und den Kernel modifizieren. Fortgeschrittene Einbrecher sind sogar in der Lage, den Kernel so zu modifizieren, dass ein installiertes Tripwire IDS anschließend den Einbruch gar nicht erkennt. Dies erfolgt durch den Austausch einiger Kernel-Routinen, die anschließend Tripwire vorgaukeln, es habe keine Modifikation des Dateisystems gegeben. Hierzu existieren bereits sogar *Rootkits*, die dies stark vereinfachen können. Derartige *Rootkits* und ihre Funktionsweise werden im Anhang besprochen.

LIDS stellt neben anderen Produkten wie GRSecurity¹, SELinux² und RSBAC³ die Lösung für dieses Problem dar. LIDS ist ein Patch des Linux-Kernels, welcher es erlaubt, die uneingeschränkten Rechte von *root* sehr feinfühlig zu administrieren, sprich zu reduzieren. Ein Einbrecher, der glaubt, *root*-Rechte zu besitzen, wird anschließend nicht in der Lage sein, die Firewall-Regeln zu modifizieren, Dienste zu beenden oder einen Sniffer zu starten. Zusätzlich erlaubt LIDS den Schutz von Dateien

1 <http://www.grsecurity.org>

2 <http://www.nsa.gov/selinux/index.cfm>

3 <http://www.rsbac.org>

und Prozessen, so dass selbst *root* sie nicht modifizieren oder entfernen darf. Hierfür verwendet LIDS ein kompliziertes Regelwerk aus Zugriffslisten ähnlich einer Firewall.

Seit Mitte 2003 verfügt LIDS über einen Lern-Modus, mit dem die Erzeugung der ACLs stark vereinfacht werden kann. Da jedoch dieser Lern-Modus auch mit Gefahren verbunden und daher mit Vorsicht einzusetzen ist, wird er in einem eigenen Abschnitt besprochen (siehe Abschnitt »LIDS Learning-Mode/ACL-Discovery« auf S. 450). Des Weiteren unterstützt LIDS Trusted Path Execution und Trusted Domain Enforcement. Diese neuen Funktionen stehen im Moment nur für den Kernel 2.6 zur Verfügung. Jedoch werden sie wahrscheinlich bis zum Erscheinen dieses Buches auch auf dem Kernel 2.4 eingesetzt werden können.

LIDS verdient dennoch seinen Namen als Intrusion-Detection-System, da es zusätzlich bei einer Verletzung des Regelwerkes einen Alarm auslösen kann. So wird ein Einbrecher, der gegen die Regeln zu verstoßen versucht, von LIDS erkannt. LIDS ist aber auch in der Lage, das entsprechende Terminal zu beenden und so den Einbrecher direkt zu stoppen.

Weitere Funktionen von LIDS sind ein Portscan-Detektor im Kernel und die Möglichkeit des automatischen E-Mail-Versands durch LIDS bei einer Regelverletzung.

In diesem Kapitel wird die Installation (Kernel 2.4) und Konfiguration von LIDS erläutert. Da die Konfiguration nicht offensichtlich ist, werden am Ende vier Beispielkonfigurationen besprochen:

- Ein Webserver basierend auf apache
- Ein DNS-Server basierend auf bind
- Ein Proxy-Server basierend auf squid
- Eine Firewall/Snort Sensor

11.2 Geschichte

Das Projekt LIDS wurde im Oktober 1999 von Huagang Xie gestartet. Im Dezember 1999 trat Philippe Biondi dem Projekt bei. LIDS wurde zunächst als Patch für den Kernel 2.2 entwickelt. Die Versionsnummern folgen diesem Muster: 0.x.x. Alle Kernel 2.2 LIDS-Patches besitzen eine führende 0. Im Herbst 2000 wurde die Entwicklung eines Kernel-Patches für den Linux-Kernel 2.3/2.4 begonnen. Diese Patches trugen die Versionsnummer 1.x.x, um sie von den Kernel 2.2-Patches zu unterscheiden. Mit Erscheinen des Kernels 2.5/2.6 wurde LIDS auch auf diesen portiert. Die entsprechenden Patches tragen nun die Versionsnummer 2.x.x.

Auf dem Kernel 2.6 nutzt LIDS für die Einbindung das Linux Security Module (LSM). Dies soll die Einbindung von LIDS in den Kernel stark vereinfachen, indem auf ein standardisiertes Interface zugegriffen wird (LSM). Diese Schnittstelle ist Bestandteil des Kernels 2.6.

11.3 Lizenz

LIDS wird von Anfang an unter der GNU General Public License (GPL) veröffentlicht. Dies erlaubt den Einsatz von LIDS in privaten wie kommerziellen Umgebungen. Die GNU GPL ist im Anhang zur Referenz abgedruckt.

11.4 Installation

Da die Installation von LIDS in Abhängigkeit vom Kernel unterschiedlich verläuft, soll hier kurz die Installation am Beispiel des Linux-Kernels 2.4.27 und 2.6.7 dargestellt werden.

Der Linux-Kernel 2.4 ist im Moment der Standard-Kernel. Jedoch wechseln die großen Distributionen eben auf den Kernel 2.6. So enthalten SUSE 9.1 und Fedora Core 2 bereits den Linux-Kernel 2.6.

11.4.1 Kernel 2.4

Dieser Abschnitt bespricht die Installation von LIDS basierend auf einem originalen Kernel 2.4. Als Beispiel wird der angesprochene Kernel 2.4.27 verwendet.

Zunächst wird der Kernel-Quelltext benötigt. Leider nimmt der LIDS-Patch derartig starke Veränderungen am Kernel vor, dass ein bereits von einem Distributor veränderter Quelltextbaum meist nicht mehr verwendbar ist. Der LIDS-Patch und die vom Distributor eingefügten Patches schließen sich meist gegenseitig aus. Das LSM-Modul (siehe nächster Abschnitt) wird hoffentlich in Zukunft für Abhilfe sorgen.

Das Quelltextarchiv kann am besten direkt vom Linux Kernel Repository (<http://www.kernel.org>) bezogen werden.

Zusätzlich wird der LIDS-Patch benötigt. Dieser steht unter der URL <http://www.lids.org> zum Download zur Verfügung.

Nach Extraktion des Kernels

```
$ cd /usr/local/src
$ tar -xvzf /path/linux-2.4.27.tar.gz # Wenn es ein tar.gz Archiv ist
$ tar -xvjf /path/linux-2.4.27.tar.bz2 # Wenn es ein tar.bz2 Archiv ist
```

kann er gepatcht werden. Dazu sind folgende Schritte erforderlich:

```
$ mv linux-2.4.27 linux-2.4.27-lids
$ tar xvzf lids-1.1.2p5-2.4.27.tar.gz
$ cd linux-2.4.27-lids
$ patch -p1 <../lids-1.1.2p5-2.4.27/lids-1.1.2p5-2.4.27.patch
```

Der Patch-Befehl sollte die Namen der gepatchten Dateien auf dem Bildschirm ausgeben. Steht der Cursor einfach auf der nächsten Zeile, ohne dass etwas passiert, ist wahrscheinlich das <-Zeichen zur Eingabeumleitung vergessen worden.

Nun muss der Kernel konfiguriert und übersetzt werden. Besitzen Sie bereits einen übersetzten Linux-Kernel, so können Sie dessen Konfiguration übernehmen. Diese ist abgespeichert unter: `/usr/src/linux/.config` oder `/boot/config-*` (Red Hat) oder unter `/boot/vmlinuz.config` (SUSE). Dazu kopieren Sie die Datei in das neue Kernel-Verzeichnis und nennen sie `.config`. Anschließend geben Sie den Befehl `make oldconfig` ein. Dieser wird die alte Konfiguration übernehmen. Anschließend können Sie die LIDS-Konfiguration anpassen.

Wenn Sie keine alte Konfigurationsdatei besitzen, müssen Sie die Konfiguration von Hand durchführen. Rufen Sie dazu entweder `make menuconfig` (auf der Textkonsole) oder `make xconfig` (auf der grafischen Oberfläche) auf.

Folgende Konfigurationseinstellungen bietet Ihnen LIDS an. Die angegebenen Werte stellen sinnvolle Startwerte für Sie dar. Einige Parameter sind selbsterklärend. Die meisten von ihnen werden im Weiteren erläutert.

```
* Linux Intrusion Detection System
*
Linux Intrusion Detection System support (EXPERIMENTAL) (CONFIG_LIDS) [Y/n/?]
*
* LIDS features
*
  Maximum protected objects to manage (CONFIG_LIDS_MAX_INODE) [256]
  Maximum ACL subjects to manage (CONFIG_LIDS_MAX_SACL) [256]
  Maximum ACL objects to manage (CONFIG_LIDS_MAX_OACL) [256]
  Hang up console when raising a security alert (CONFIG_LIDS_HANGUP) [N/y/?]
  Security alert when execing unprotected programs before sealing LIDS
↳ (CONFIG_LIDS_SA_EXEC_UP) [N/y/?]
  Attempt not to flood logs (CONFIG_LIDS_NO_FLOOD_LOG) [Y/n/?]
  Authorised time between two identic logs (seconds)
↳ (CONFIG_LIDS_TIMEOUT_AFTER_FLOOD) [60]
  Allow switching LIDS protections (CONFIG_LIDS_ALLOW_SWITCH) [Y/n/?]
  Implicitly protect LIDS admin passwd (CONFIG_LIDS_PROTECT_PWD) [N/y/?]
  Restrict mode switching to specified terminals
↳ (CONFIG_LIDS_RESTRICT_MODE_SWITCH) [N/y/?]
  Number of attempts to submit password (CONFIG_LIDS_MAX_TRY) [3]
  Time to wait after a fail (seconds) (CONFIG_LIDS_TTW_FAIL) [3]
  Allow any program to switch LIDS protections
↳ (CONFIG_LIDS_ALLOW_ANY_PROG_SWITCH) [N/y/?]
  Allow reloading config. file (CONFIG_LIDS_RELOAD_CONF) [Y/n/?]
  Port Scanner Detector in kernel (CONFIG_LIDS_PORT_SCAN_DETECTOR) [Y/n/?]
  Send security alerts through network (CONFIG_LIDS_SA_THROUGH_NET) [N/y/?]
  LIDS Debug (CONFIG_LIDS_DEBUG) [N/y/?]
```

Wenn ein `make`-Befehlsaufruf fehlschlägt, prüfen Sie bitte zunächst, ob Sie sich im richtigen Verzeichnis befinden. Der Befehl sucht nach einer Datei *Makefile*, die sich im neu angelegten Kernel-Verzeichnis befindet. Das bedeutet, dass das Kernel-Quelltext-Verzeichnis ihr aktuelles Arbeitsverzeichnis sein muss. Wenn das nicht der Grund des Fehlers ist, prüfen Sie, ob alle Pakete zur Übersetzung des Kernels installiert wurden. Da dies distributionsabhängig ist, schauen Sie bitte in der Dokumentation Ihrer Distribution nach.

Anschließend wird der Kernel übersetzt mit den Befehlen:

```
$ make depend
$ make bzImage
$ make modules
```

Nun muss dieser Kernel noch installiert werden. Dazu geben Sie bitte folgende Befehle ein:

```
# make modules_install
# make install
```

Auf einigen Distributionen schlägt der letzte Befehl fehl. Dann müssen Sie den Kernel manuell installieren. Normalerweise kopiert der Befehl `make install` den Kernel in das */boot*-Verzeichnis, erzeugt die Ramdisk und aktualisiert den Bootloader. Wenn er fehlschlägt, führen Sie bitte die folgenden Befehle aus:

```
# cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.27-lids
# cp System.map /boot/System.map-2.4.27
```

Möglicherweise muss nun noch eine initiale Ramdisk erzeugt werden. Dies erfolgt auf SUSE mit dem Befehl `mk_initrd -k "vmlinuz-2.4.27-lids"` und unter Red Hat mit dem Befehl `mkinitrd /boot/initrd-2.4.27 2.4.27`. Wenn Sie sich nicht sicher sind, so erzeugen Sie bitte diese Ramdisk.

Damit nun der Boot-Manager den Kernel beim nächsten Boot-Vorgang auch findet, muss er ihm bekannt gegeben werden. Zwei verschiedene Boot-Manager finden momentan Gebrauch: Lilo und Grub.

Zur Konfiguration von Lilo editieren Sie bitte die Datei */etc/lilo.conf* und hängen die folgenden Zeilen an (ersetzen Sie bitte *Ihr Rootdevice* mit dem entsprechenden Gerät, kopieren Sie den Eintrag einfach):

```
image = /boot/vmlinuz-2.4.27-lids
label = lids
root = /dev/Ihr Rootdevice
initrd = /boot/initrd-2.4.27-lids
```

Anschließend rufen Sie bitte einmal den Befehl `lilo` auf.

Zur Konfiguration von Grub fügen Sie bitte die folgenden Zeilen an die Datei `/etc/grub.conf` an:

```
title Lids (2.4.27-lids)
    root (<Ihr Bootdevice>)
    kernel /vmlinuz-2.4.27-lids ro root=/dev/<Ihr Rootdevice>
    initrd /initrd-2.4.27-lids.img
```

Ein Aufruf von Grub ist *nicht* nötig.

Damit wurde der Kernel installiert und nun können die Administrationswerkzeuge übersetzt und installiert werden.

Zunächst muss der Quelltext der Administrationswerkzeuge konfiguriert werden. Dies erfolgt mit dem Werkzeug `configure`:

```
$ cd lids-1.1.2p5-2.4.27/lidstools-0.4.3p1/
$ ./configure KERNEL_DIR=/usr/local/src/linux-2.4.27-lids
$ make
# make install
```

Das Installationsscript der Version 0.x.x oder 1.x.x verlangt die Eingabe eines Kennwortes. Bitte merken Sie sich das von Ihnen eingegebene Kennwort gut. Es stellt die einzige Möglichkeit dar, LIDS später online zu administrieren. Anschließend ist lediglich ein Neustart erforderlich. Denken Sie daran, beim Neustart im Boot-Menü den neuen Kernel auszuwählen. Funktioniert dieser, so können Sie ihn auch als Default-Kernel einstellen:

- Bei Lilo tragen Sie dazu folgende Zeile in der Datei `/etc/lilo.conf` ein (oder editieren die vorhandene): `default=lids`
- Grub zählt die Einträge von oben beginnend mit Null durch. Dort müssen Sie entsprechend die Zahl in der Zeile `default=X` ändern.

11.4.2 Kernel 2.6

Dieser Abschnitt bespricht die Installation von LIDS basierend auf einem Kernel 2.6 mit LSM-Modul. Als Beispiel wird der angesprochene Kernel 2.6.7 verwendet. Diese Installation ähnelt sehr stark dem letzten Kapitel. Entsprechende Punkte werden hier nicht wiederholt.

Zunächst benötigen Sie den Kernel-Quelltext. Extrahieren Sie ihn wie im letzten Abschnitt beschrieben.

```
$ cd /usr/local/src
$ tar xjf linux-2.6.7.tar.bz2
$ tar xzf lids-2.2.0rc3-2.6.7.tar.gz
$ mv linux linux-2.6.7-lids
```

```
$ cd linux-2.6.7-lids
$ patch -p1 < ../lids-2.2.0rc3-2.6.7/lids-2.2.0rc3-2.6.7.patch
```

Der Patch modifiziert automatisch in der Datei *Makefile* den Abschnitt `EXTRAVERSION =!`. Anschließend ist erneut die Konfiguration des Kernels erforderlich. Hierbei kann ebenfalls wie im vorherigen Abschnitt beschrieben auf eine alte Konfigurationsdatei zurückgeriffen werden. Bei der Konfiguration ist es wichtig, darauf zu achten, dass keine weiteren Security-Moduls gleichzeitig aktiv sind.

Folgende neuen Optionen stehen nun zur Verfügung:

```
* LIDS support
*
Linux Intrusion Detection System support (EXPERIMENTAL) (LIDS) [Y/n/m/?] y
*
* LIDS Options
*
Attempt not to flood logs (LIDS_NO_FLOOD_LOG) [Y/n/?] y
Allow switching the LFS and States (LIDS_ALLOW_SWITCH) [Y/n/?] y
  Allow switch the Linux Free Session (LIDS_ALLOW_LFS) [Y/n/?] y
  Restrict mode switching to specified terminals
  ↪ (LIDS_RESTRICT_MODE_SWITCH) [Y/n/?] y
    Allow mode switching from a Linux Console (LIDS_MODE_SWITCH_CONSOLE)
    ↪ [Y/n/?] y
    Allow mode switching from a serial Console (LIDS_MODE_SWITCH_SERIAL)
    ↪ [N/y/?] n
    Allow mode switching from a PTY (LIDS_MODE_SWITCH_PTY) [N/y/?] n
```

Die weitere Installation des Kernels und der Administrationswerkzeuge erfolgt wie im letzten Abschnitt beschrieben. Allerdings erfordert LIDS in der Version 2.x.x nicht die Eingabe eines Kennwortes.

11.5 LIDS-Hintergründe

Wie bereits in der Einleitung angesprochen wurde, stellt eines der Sicherheitsprobleme unter Linux und UNIX die Allmacht von *root* dar. Sobald es einem Einbrecher gelingt, *root*-Privilegien zu erhalten, steht ihm das gesamte System offen. Diese Gefahr ist recht früh unter UNIX und auch unter Linux erkannt worden. Zur Abhilfe wurden die einzelnen *root*-Privilegien aufgeschlüsselt in so genannte *Capabilities* (Fähigkeiten). Eine Untergruppe dieser Fähigkeiten wurde als so genannte *POSIX-Capabilities* auch standardisiert.

Damit nun ein Benutzer bzw. ein Prozess eine bestimmte Aktion ausführen kann, benötigt er die entsprechende Fähigkeit. Ein Beispiel: Der Benutzer *root* möchte die IP-Adresse der Netzwerkkarte ändern. Für diesen Vorgang benötigt er die Fähigkeit `CAP_NET_ADMIN`. Normalerweise besitzt *root* unter Linux alle Fähigkeiten. Die Gesamtheit der für *root* verfügbaren Fähigkeiten werden in dem so genannten *Capabil-*

ities Bounding Set zusammengefasst. Linux bietet bereits die Möglichkeit, einzelne Fähigkeiten aus dieser Menge zu entfernen. Leider besteht anschließend keine Möglichkeit mehr, diese Fähigkeit zurückzuerlangen, außer wenn der Rechner neu gestartet wird. Die Fähigkeit ist dauerhaft verloren. Dies stellt sicherlich einen Vorteil in Hochsicherheitsumgebungen dar, erschwert aber auch die Administration ungemein.

LIDS baut nun auf diesen Linux-*Capabilities* auf und erweitert sie. Zunächst besteht die Möglichkeit, ohne einen Neustart die entsprechenden Fähigkeiten von *root* zu aktivieren und zu deaktivieren. Dies bedeutet bereits einen starken Fortschritt. Zusätzlich bietet LIDS aber bestimmten Prozessen die Möglichkeit, auf bestimmte Fähigkeiten zuzugreifen. War es in der Vergangenheit immer nötig, dass der Befehl *su* mit *SetUID-root*-Rechten installiert werden musste, damit er über die gesamte Menge des *Bounding Sets* verfügen durfte, so genügt es nun, dem Befehl die Fähigkeiten *CAP_SETUID* und *CAP_SETGID* zuzuweisen. Entdeckt ein Einbrecher eine Sicherheitslücke in dem *su*-Befehl, die es ihm erlaubt, die Firewall zu stoppen (*CAP_NET_ADMIN*), so weist dieser Befehl *su* zwar die Sicherheitslücke, aber nicht die notwendige Fähigkeit auf. Die Firewall läuft weiter.

Diese Fähigkeiten erlauben es auch, Prozesse zu schützen. LIDS hat hierzu weitere Fähigkeiten implementiert, die nicht üblicherweise unter Linux zur Verfügung stehen. So existieren die Fähigkeiten *CAP_HIDDEN*, *CAP_PROTECTED* und *CAP_KILL_PROTECTED*. Diese Fähigkeiten verstecken einen Prozess, schützen ihn vor Signalen (z.B. *kill -9*) und erlauben es einem Prozess, geschützte Prozesse dennoch zu beenden.

Da LIDS die Modifikationen im Kernel durchführt, existiert keine Möglichkeit aus dem Userspace, diese Modifikationen zu umgehen. Jedoch existieren bereits einige weitere Kernel-Module, die von Einbrechern geschrieben wurden. Diese Module versuchen genau die erwähnten Sicherheiten auszuhebeln. Kernelbasierte Rootkits existieren seit etwa 4-5 Jahren. Im Jahr 2001 wurde das Kernel-Intrusion-System (KIS) vorgestellt. Dieses System ist in der Lage, den Kernel direkt durch den Zugriff über das Gerät */dev/kmem* zu modifizieren. Genauere Erläuterungen sind im Anhang zu finden.

In der Vergangenheit wurde hier häufig die Verwendung eines Kernels ohne Modulunterstützung empfohlen. Kernelbasierte *Rootkits* benutzten in der Vergangenheit häufig ladbare Kernel-Module, (LKM) um ihre Funktionalität im Kernel zu implementieren. Seit der Verfügbarkeit des *Rootkits* KIS ist dies jedoch kein Schutz mehr. KIS ist in der Lage, selbst bei einem monolithischen Kernel Veränderungen über einen direkten Schreibzugriff auf den Arbeitsspeicher (*/dev/kmem*) durchzuführen. Außerdem erzeugt ein monolithischer Kernel zusätzlichen Administrationsaufwand, wenn Hardware ausgetauscht werden soll. Meist ist die entsprechende Unterstützung nicht im Kernel aktiviert worden. Eine Neuübersetzung ist erforderlich. Bei einem modularen Kernel ist lediglich das Modul auszutauschen. LIDS ist in der Lage, einen modularen Kernel so zu schützen, dass sowohl mit LKMs als auch über */dev/kmem* kein Einbruch in den Kernel möglich ist. Auf dem Gerät */dev/kmem* darf nicht mehr geschrieben werden.

Es ist daher notwendig, dass der Kernel vor derartigen Angriffen geschützt wird. LIDS ist in der Lage, dies zu tun. Dazu wird nach dem Start sämtlicher Dienste und dem Laden sämtlicher für den Betrieb benötigter Kernel-Module der Kernel versiegelt. Anschließend sind alle Beschränkungen aktiviert und das Laden bzw. Entfernen von Kernel-Modulen wird unterbunden (CAP_SYS_MODULE). Ein Zugriff auf `/dev/kmem` oder direkt auf die Festplatten (z.B. `/dev/hda`) kann und soll über CAP_SYS_RAWIO unterbunden werden. Anschließend befindet sich das System bei korrekter Konfiguration in einem sehr sicheren Zustand.

Es besteht jedoch weiterhin die Gefahr, dass ein Einbrecher wesentliche Dateien des Rechners modifiziert oder Befehle austauscht. LIDS bietet hier eine Tripwire ähnliche Funktionalität auf Kernel-Ebene. Jede einzelne Datei oder ganze Verzeichnisse können von LIDS so überwacht werden, dass kein Zugriff, kein lesender Zugriff, kein nur anhängender Zugriff oder kein schreibender Zugriff erlaubt wird. Dies kann erlaubt werden in Abhängigkeit vom zugreifenden Befehl und von der aktuellen Uhrzeit. Anwendungsbeispiele folgen weiter unten. LIDS ersetzt jedoch Tripwire nicht. Die Überwachung mit LIDS erfordert einen höheren administrativen Aufwand, wenn Modifikationen speziell gewünscht werden. Teilweise erlaubt LIDS auch einigen Prozessen die Modifikation. Möglicherweise findet der Einbrecher genau diese Möglichkeit. Die von den Prozessen benötigten Fähigkeiten werden ähnlich verwaltet. Auch hier besteht die Möglichkeit, einem Prozess eine bestimmte Fähigkeit zu einem bestimmten Zeitpunkt zuzuweisen.

Die gesamte Verwaltung des Dateizugriffes und der Fähigkeiten erfolgt auf der Dateisystemebene. Hierbei wird von LIDS die Virtuelle Dateisystemebene (VFS) gewählt. Dies ermöglicht den Einsatz auf jedem Dateisystem (`ext2`, `ext3`, `reiserfs`, ...). Da LIDS zur internen Verwaltung die Inodes verwendet, ist die Wahl des Begriffes Prozess ein wenig irreführend. Soll zum Beispiel der Webserver Prozess das Recht erhalten, Daten an die Datei `/var/log/httpd/access.log` anzuhängen, so erhält der Inode, der den Befehl `httpd` beherbergt, das entsprechende Recht. Dies führt zu Problemen, wenn zum Beispiel die Datei `/var/log/httpd/access.log` rotiert wird. Auch diese Datei wird über ihren Inode in den Zugriffslisten referenziert. Eine Rotation benennt jedoch meist die vorhandene Datei (z.B. `/var/log/messages`) lediglich um (`/var/log/messages.1`). Das bedeutet, dass diese Datei weiterhin in demselben Inode abgespeichert ist. Die Protokollierung wird in einer neuen Datei mit neuem Inode wieder aufgenommen. LIDS überwacht jedoch weiterhin die alte Datei. Dieses Problem kann umgangen werden, indem die Rotation deaktiviert, LIDS nach der Rotation neu initialisiert wird oder nach der Rotation die Protokollierung weiterhin im originalen Inode vorgenommen wird (Option `copy-truncate`). Letztere Funktion wird nicht von allen Rotationswerkzeugen unterstützt. Hierbei wird die Protokolldatei kopiert und zur Rotation umbenannt. Anschließend wird der Datei-Inhalt des originalen Inodes geleert und die weitere Protokollierung hier vorgenommen.

11.6 Konfiguration

Wenn die Installation von LIDS mit dem LIDS-Installationsskript vorgenommen wurde (`make install`), dann befindet sich bereits auf Ihrem Linux-System eine funktionsfähige LIDS-Konfiguration, die lediglich noch angepasst werden muss. Diese Anpassung soll später besprochen werden. Zunächst werden jedoch die Werkzeuge, deren Syntax und Konfigurationsdateien besprochen. Dies wird an einigen kleinen Beispielen vollzogen, bevor dann die eigentliche Konfiguration für einen vollständigen Linux-Server besprochen wird.

Das Kapitel LIDS wird mit einer Betrachtung der sinnvollerweise durch LIDS zu schützenden Rechner abschließen.

11.6.1 Kommandozeilenwerkzeuge

Abhängig von den LIDS-Versionen stehen ein oder zwei verschiedene Werkzeuge zur Verfügung. Frühe LIDS-Versionen kannten nur das Werkzeug *lidsadm*. Mit diesem Werkzeug wurde die gesamte Konfiguration und Administration durchgeführt. Moderne LIDS-Versionen (ab Version 1.1) verwenden zwei verschiedene Werkzeuge: *lidsadm* und *lidsconf*. *lidsconf* ist das Konfigurationswerkzeug, mit dem die Zugriffsregeln gesetzt und modifiziert werden. *lidsconf* speichert diese Informationen in der Datei `/etc/lids/lids.conf`. *lidsadm* ist das Administrationswerkzeug, welches verwendet wird, um LIDS zu aktivieren, zu deaktivieren etc.

lidsadm

Der Befehl `lidsadm` ist das zentrale Administrationswerkzeug für LIDS. Mit diesem Werkzeug ist es möglich, im laufenden Betrieb den Kernel zu versiegeln und LIDS bzw. einzelne Fähigkeiten an- oder abzuschalten.

Hierzu unterstützt der Befehl folgende Funktionen:

- -h. Anzeige der Hilfe
- -I. Versiegeln des Kernels
- -S. An- bzw. Abschalten von LIDS oder einer Fähigkeit. Hierbei muss das LIDS-Kennwort angegeben werden.
 - (+|-) KEY. An- (+) oder abzuschaltende (-) Fähigkeit (siehe unten)
- -V. Anzeigen des LIDS-Status

Am wichtigsten ist die Versiegelung des Kernels. Diese sollte erfolgen, sobald das System seinen Arbeitszustand erreicht hat, sämtliche Dienste gestartet und sämtliche für ihre Funktion erforderlichen Module geladen wurden. Daher kann der entsprechende Befehl gut in Form eines Startscriptes eingebunden werden.

Im Folgenden ist ein Beispielscript für Red Hat Linux abgedruckt, welches dann in die Startsequenz eingebaut werden kann:

```
#!/bin/bash
#
# lids          This scripts seals the Kernel.
#
# chkconfig: 345 99 99
# description: This calls lids and seals the Kernel. No module may be loaded\
#              afterwards.

case "$1" in
  start)
    action /usr/sbin/lidsadm -I
    ;;
esac
```

Natürlich ist es ebenfalls möglich, den Aufruf in ein `rc.local`-Script einzubauen.

lidsconf

Der Befehl `lidsconf` ist das zentrale Konfigurationswerkzeug für LIDS. Mit diesem Werkzeug ist der Administrator in der Lage, die zu überwachenden Dateien und Prozesse mit ihren Eigenschaften anzugeben. `lidsconf` speichert diese Konfiguration in der Datei `/etc/lids/lids.conf` (siehe unten) ab.

Die Verwaltung der Regeln erfolgt ähnlich der Verwaltung von Firewallregeln mit `ipchains` oder `iptables`. Folgende Optionen stehen zur Verwaltung der Regeln zur Verfügung:

- -A. Füge eine Regel hinzu (Add).
- -D. Entferne eine Regel (Delete).
- -Z. Lösche alle Regeln (Zero).
- -U. Aktualisiere die Inode-Nummern in der Konfigurationsdatei (Update).
- -L. Zeige die Regeln an (List).
- -P. Definiere ein Passwort und speichere es RipeMD-160-verschlüsselt.
- -v. Zeige die Versionsnummer.
- -h. Zeige die Hilfe.

Die Syntax der Regeln wird weiter unten besprochen. Wichtig ist an dieser Stelle der Hinweis, dass sich die Regeln immer auf Inode-Nummern oder Gerätenummern beziehen. Ändert sich der Name der Datei, so wird LIDS immer noch die Datei schützen. Wird eine neue Datei mit identischem Namen angelegt und die alte Datei gelöscht, so wird LIDS weiterhin die alte (gelöschte) Datei schützen. LIDS müssen derartige Veränderungen mit der Option `-U` bekannt gemacht werden. Es wird dann seine gesamte Inodetabelle neu aufbauen.

LIDS-Fähigkeiten

Die Privilegien von *root* und der einzelnen Prozesse sind in Fähigkeiten aufgeteilt. Einige dieser Fähigkeiten stehen unter jedem Linux-System zur Verfügung, einige weitere wurden durch den LIDS-Kernel-Patch hinzugefügt. Die Fähigkeiten werden im Kernel durch Nummern repräsentiert. Im Folgenden werden die Fähigkeiten in dieser numerischen Reihenfolge aufgeführt und erklärt.

1. CAP_CHOWN. Fähigkeit, die Befehle `chown` und `chgrp` zu verwenden
2. CAP_DAC_OVERRIDE. Kompletter Zugriff auf Verzeichnisse und Dateien unabhängig von den gesetzten Rechten (Discretionary Access Control Override)
3. CAP_DAC_READ_SEARCH. Lese- und Suchzugriff auf Verzeichnisse und Dateien unabhängig von den gesetzten Rechten
4. CAP_FOWNER. Fähigkeit, Änderungen an einer Datei eines anderen Benutzers durchzuführen
5. CAP_FSETID. Fähigkeit, die SetUID- und SetGID-Rechte an Dateien anderer Benutzer zu setzen
6. CAP_KILL. Fähigkeit, Signale an Prozesse anderer Benutzer zu senden
7. CAP_SETGID. Erlaubt ein SetGID
8. CAP_SETUID. Erlaubt ein SetUID
9. CAP_SETPCAP. Erlaubt eine Weitergabe der eigenen Fähigkeiten an andere Prozesse
10. CAP_LINUX_IMMUTABLE. Erlaubt die Modifikation der Attribute `append-only` und `immutable`
11. CAP_NET_BIND_SERVICE. Erlaubt die Verwendung von TCP/UDP-Ports <1024
12. CAP_NET_BROADCAST. Erlaubt die Verwendung von Broadcast und Multicast
13. CAP_NET_ADMIN. Erlaubt viele Netzwerkadministrationsvorgänge (Netzwerkkarten, Firewall, Routing, Promiscuous Modus etc.)
14. CAP_NET_RAW. Erlaubt die Verwendung von RAW sockets (Ping)
15. CAP_IPC_LOCK. Erlaubt ein Locking von gemeinsam genutzten Speicherbereichen
16. CAP_IPC_OWNER. Übergeht InterProcessCall-Besitzbeschränkungen
17. CAP_SYS_MODULE. Erlaubt das Laden und Entfernen von Kernel-Modulen
18. CAP_SYS_RAWIO. Erlaubt den direkten Zugriff auf Geräte (`/dev/hda`, `/dev/kmem`)
19. CAP_SYS_CHROOT. Erlaubt die Verwendung von `chroot`
20. CAP_SYS_PTRACE. Erlaubt die Prozessverfolgung (z.B. mit `strace`)
21. CAP_SYS_PACCT. Erlaubt die Konfiguration von Prozess-Accounting (`lastcomm`)
22. CAP_SYS_ADMIN. Erlaubt viele Administrationsvorgänge (`hostname`, `mount`, `quota`, `swap`, `dma` etc.)
23. CAP_SYS_BOOT. Erlaubt den Reboot des Kernels
24. CAP_SYS_NICE. Erlaubt die Erhöhung der Priorität eigener und anderer Prozesse
25. CAP_SYS_RESOURCE. Erlaubt das Setzen und Übergehen von Ressourcengrenzen

- 26. CAP_SYS_TIME. Erlaubt das Setzen der Rechnerzeit (sowohl System- als auch CMOS-Zeit)
- 27. CAP_SYS_TTY_CONFIG. Erlaubt die Konfiguration von TTYs (z.B. stty)
- 28. CAP_MKNOD. Erlaubt die Erzeugung von Gerätedateien
- 29. CAP_LEASE. Erlaubt das Setzen von Leases (filelocking read/write)

LIDS hat die folgenden Fähigkeiten hinzugefügt:

- 30. CAP_HIDDEN. Der Prozess wird vor dem gesamten System versteckt (kein Eintrag in */proc*)
- 31. CAP_KILL_PROTECTED. Der Prozess hat das Recht, einen geschützten Prozess zu beenden.
- 32. CAP_PROTECTED. Der Prozess ist geschützt. Lediglich SIG_CHLD-Signale von Kindprozessen und Signale von CAP_KILL_PROTECTED-Prozessen sind erlaubt.

Der Befehl `lidsadm` unterstützt zusätzlich noch folgende Schlüssel:

- 33. LIDS_GLOBAL. An- und Abschalten von LIDS für den gesamten Rechner
- 34. RELOAD_CONF. Erneutes Einladen der Konfigurationsdateien
- 35. LIDS. An- und Abschalten von LIDS für die aktuelle Session. Dies bezeichnet man als LIDS-free Session (LFS). Die aktuelle Shell und alle gestarteten Kindprozesse stehen nicht unter der Überwachung durch LIDS.

11.6.2 LIDS 1.x.x Konfigurationsdateien

LIDS verwendet in den Versionen 0.x.x und 1.x.x vier verschiedene Konfigurationsdateien: *lids.conf*, *lids.cap*, *lids.net* und *lids.ps*. Diese Dateien befinden sich im Verzeichnis */etc/lids*. Dieses Verzeichnis wird von LIDS automatisch versteckt, wenn LIDS aktiv ist.

lids.conf

Die Datei */etc/lids/lids.conf* enthält die Regeln, die mit dem Werkzeug `lidsconf` erstellt wurden. `lidsconf` ist im Wesentlichen ein Werkzeug, um die Regeln in dieser Datei zu administrieren. Diese Datei wird verwendet, um bei einem Neustart des Systems diese Regeln einlesen zu können. Darüber hinaus muss nach jeder Administration mit `lidsconf` der Inhalt dieser Datei neu eingelesen werden. Dies erfolgt mit dem Befehl `lidsadm -S -- +RELOAD_CONF` (siehe unten). Damit ein derartiges Neueinlesen erlaubt ist, muss LIDS mit der Option *Allow reloading configuration file (CONFIG_LIDS_RELOAD_CONF) [Y/n/?] y* im Kernel konfiguriert worden sein.

Der Inhalt der Datei ähnelt der folgenden Liste:

```
#
# This file is auto generated by lidsconf
# Please do not modify this file by hand
#
```

```
0:0::1:0:402491:2050:/sbin:0-0
0:0::1:0:402486:2050:/bin:0-0
0:0::1:0:2:2049:/boot:0-0
0:0::1:0:402490:2050:/lib:0-0
0:0::1:0:305825:2050:/usr:0-0
0:0::1:0:225345:2050:/etc:0-0
```

Diese Datei wird automatisch von *lidsconf* generiert und sollte nicht von Hand angepasst werden. Dennoch soll das Format der Datei kurz erklärt werden. Die Datei enthält eine Tabelle, deren Spalten durch Doppelpunkte (:) getrennt werden.

```
s_inode:s_dev:s_file:Typ:Vererbung:o_inode:o_dev:o_file:von-bis
```

Die erste Spalte (*s_inode*) enthält den Inode des Subjekts der Operation. Die zweite Spalte (*s_dev*) enthält das Gerät, auf dem sich der Inode befindet. Die dritte Spalte (*s_file*) enthält den Dateinamen des Subjektes. Das Subjekt ist der Prozess, der die Operation durchführt (s.u.). Wurde kein Subjekt definiert, so werden die Spalten mit 0 gefüllt bzw. freigelassen.

Die vierte Spalte (Typ) enthält den Typ des Zugriffes. Hierbei sind die folgenden Zahlen den Typen zugeordnet: 0-Deny, 1-Readonly, 3-Append, 7-Write und 16-Grant. In der fünften Spalte wird die Vererbung der Rechte auf Kindprozesse definiert. Hier können die Ebenen angegeben werden (-1 = unendlich).

Die Spalten 6, 7 und 8 enthalten den Inode, das Gerät und den Dateinamen des Objektes. Handelt es sich bei dem Objekt um eine Fähigkeit, so wird die Nummer der Fähigkeit als Gerät eingetragen.

Die letzte Spalte definiert schließlich den Zeitraum, währenddessen die Regel aktiv ist.

LIDS prüft bei jedem Einlesen die Syntax dieser Datei. Sollten Sie manuell Änderungen vornehmen und dabei Syntaxfehler einfügen, so wird LIDS diese Fehler melden.

lids.cap

Die Datei */etc/lids/lids.cap* definiert den Satz der Fähigkeiten, auf die der Kernel zugreifen kann. Sämtliche Prozesse, selbst solche, die mit *root*-Privilegien laufen, können maximal nur auf diesen Satz zugreifen. Der Inhalt dieser Datei ist der folgende:

```
-0:CAP_CHOWN
+1:CAP_DAC_OVERRIDE
+2:CAP_DAC_READ_SEARCH
+3:CAP_FOWNER
+4:CAP_FSETID
-5:CAP_KILL
```

Auch diese Datei ist als Tabelle aufgebaut. Sie enthält zwei Spalten durch Doppelpunkt (:) getrennt. In der ersten Spalte befindet sich die Nummer der Fähigkeit. Dieser Nummer wird ein Minus (-) vorangestellt, um die Fähigkeit abzuschalten, und ein Plus (+), um die Fähigkeit anzuschalten. In der zweiten Spalte wird der Name der Fähigkeit eingetragen. Diese Datei kann nicht mit `lidsconf` administriert werden. Die Einstellungen in dieser Datei sind aktiv, sobald der Kernel versiegelt wird. Dann liest `lidsadm` die Datei aus und setzt sie im Kernel um. Benötigt ein Prozess später eine Capability, die sich nicht in dieser Datei befindet, so muss sie ihm spezifisch zugewiesen werden (s.u. GRANT).

Aus Sicherheitsgründen sollten möglichst viele Fähigkeiten deaktiviert werden.

lids.pw

Die Datei `/etc/lids/lids.pw` enthält das mit RipeMD-160 verschlüsselte Kennwort von LIDS. RipeMD-160 ist eine kryptografische Hash-Funktion, die einen 160-Bit-Wert erzeugt. RipeMD-160 stellt damit einen längeren Hash als MD5, welches lediglich einen 128-Bit-Hash erzeugt, zur Verfügung. RipeMD-160 fällt unter keine Patente und ist frei verfügbar.

Das Kennwort wird mit dem Befehl `lidsconf -P` in der Datei erzeugt.

lids.net

Die Datei `/etc/lids/lids.net` enthält die Konfigurationseinstellungen für den Versand von E-Mails. LIDS ist in der Lage, Regelverletzungen per E-Mail direkt zu melden. Dazu muss der Kernel mit der entsprechenden Option konfiguriert worden sein (s.o.). Ist dies der Fall, so versendet LIDS, wenn die Option `MAIL_SWITCH=1` gesetzt ist, eine E-Mail für jede Regelverletzung. Wenn der E-Mail-Versand zunächst getestet werden soll, so kann der Kernel entsprechend konfiguriert und anschließend mit dieser Option der Versand an- bzw. abgeschaltet werden. Hier wird auszugsweise die Standarddatei gezeigt.

```
MAIL_SWITCH= 1

MAIL_RELAY=127.0.0.1:25

MAIL_SOURCE=lids.sinocluster.com

MAIL_RELAY=127.0.0.1:25

MAIL_FROM= LIDS_ALERT@lids.sinocluster.com

MAIL_TO= root@localhost

MAIL_SUBJECT= LIDS ALert
```

11.6.3 LIDS 2.x.x Konfigurationsdateien

LIDS verwendet in der Version 2.x.x sieben verschiedene Konfigurationsdateien: *lids.ini* und für jeden der Zustände *boot*, *postboot* und *shutdown* eigene *lids*.conf*- und *lids*.cap*-Dateien. Diese Dateien befinden sich im Verzeichnis */etc/lids*. Dieses Verzeichnis wird von LIDS automatisch versteckt, wenn es aktiv ist.

lids.ini

Diese Datei enthält im Moment im Grunde nur eine Einstellung: `ACL_DISCOVERY`. Hiermit können Sie den Lern-Modus in LIDS aktivieren. Dieser Modus wird in Abschnitt »LIDS Learning-Mode/ACL-Discovery« auf S. 450 besprochen.

```
#
# This file contain some initial value for LIDS
#
# ACL_DISCOVERY =1 # will turn on the acl discovery
# ACL_DISCOVERY =0 # will turn off the acl discovery
#
ACL_DISCOVERY=0
# end of this file,
# DO NOT DELETE THIS LINE
```

lids*.conf und lids*.cap

In der Version 2.x.x kann LIDS verschiedene Betriebszustände unterscheiden und hier unterschiedliche Regelsätze anwenden. LIDS kennt die folgenden drei Zustände:

- **BOOT:** Dies ist der Zustand, in dem sich der Rechner automatisch nach dem Boot des Kernels befindet. Dieser Zustand wird genutzt, um den Rechner zu initialisieren und alle benötigten Dienste zu starten.
- **POSTBOOT:** Dies ist der normale Zustand, in dem der Rechner seine Alltagsaufgaben erledigt. LIDS erreicht diesen Zustand durch den Befehlsaufruf: `lidsadm -I`. Dieser Befehl versiegelt den Kernel und wechselt in den *POSTBOOT*-Zustand.
- **SHUTDOWN:** Dieser Zustand erlaubt ein Herunterfahren des Rechners. Dienste dürfen in diesem Zustand beendet und Dateisysteme ausgehängt werden. Sie wechseln mit `lidsadm -S -- +SHUTDOWN` in diesen Zustand.

Für jeden dieser Zustände gibt es nun eigene *.cap*- und *.conf*-Dateien, die die Regeln und Rechte für diese Zustände definieren. Zusätzlich können noch globale ACLs und Capabilities definiert werden. Diese werden dann in den Dateien *lids.conf* und *lids.cap* in dem Verzeichnis */etc/lids* gespeichert. Dabei kann es natürlich zu Konflikten kommen, da entgegengesetzte Regeln in den globalen und den für den Zustand spezifischen Dateien gespeichert sein können. Um derartige Konflikte zu finden,

können Sie den Befehl `lidsconf -C` verwenden. Dieser überprüft die Integrität der Konfiguration und gibt jede Form von Warnung aus, so auch Warnungen, die nicht mit Konflikten zwischen den verschiedenen Dateien zu tun haben:

```
/bin/su GRANT: 0 CAP_SETUID
lidsconf: You must protect the subject file /bin/su or its directory as
↳ READONLY or DENY.
```

Um nun eine ACL in einem bestimmten Zustand hinzuzufügen, geben Sie bei dem `lidsconf`-Aufruf den Zustand an:

```
lidsconf -A BOOT -s subject -o object -j action
```

Direkt nach der Installation sind bereits einige rudimentäre Dateien vorhanden. So ist die folgende Datei für *BOOT* zuständig:

```
#
# This file is auto generated by lidsconf
# Please do not modify this file by hand
#
0:0::1:0:16610:770:/sbin/depmod:0-0
0:0::1:0:288004:770:/lib:0-0
16610:770:/sbin/depmod:7:0:288004:770:/lib:0-0
```

Diese ACLs schützen den Befehl `/sbin/depmod` und das Verzeichnis `/lib`. Im Zustand *POSTBOOT* werden die folgenden ACLs per Default geladen:

```
#
# This file is auto generated by lidsconf
# Please do not modify this file by hand
#
0:0::1:0:224001:770:/etc:0-0
96627:770:/bin/su:16:0:-1:7:CAP_SETUID:0-0
96627:770:/bin/su:16:0:-1:6:CAP_SETGID:0-0
320557:770:/usr/sbin/sshd:16:0:-1:6:CAP_SETGID:0-0
320557:770:/usr/sbin/sshd:1:0:228618:770:/etc/shadow:0-0
320557:770:/usr/sbin/sshd:16:0:-1:7:CAP_SETUID:0-0
96627:770:/bin/su:1:0:228618:770:/etc/shadow:0-0
```

Hier wird das Verzeichnis `/etc` geschützt. Die Befehle `/bin/su` und `/usr/sbin/sshd` erhalten hier besondere Rechte für den Zugriff auf die Datei `/etc/shadow`. Diese ACLs sind natürlich lange nicht ausreichend für einen Schutz des Rechners. Weiter unten werden weitere Beispiele besprochen, die entweder hier oder in der globalen Konfigurationsdatei eingetragen werden sollten. Im *SHUTDOWN*-Zustand schließlich werden noch keine besonderen ACLs aktiviert.

Sie können eigentlich ohne Änderung die alten ACLs als globale übernehmen. Sinnvollerweise passen Sie diese aber auf die verschiedenen Zustände an.

11.6.4 Regelsyntax und Beispiele

Syntax

Die Regeln, die mit `lidsconf` definiert werden können, haben folgende allgemeine Syntax:

```
[ -s subject ] -o object [ -d ] [ -t from-to ] [ -i level ] -j ACTION
```

- `-s subject`. Ausführendes Programm auf dem System, z.B. `/bin/su`
- `-o object`. Datei, Verzeichnis oder Fähigkeit, auf das der Zugriff erfolgt. Falls es sich um `CAP_NET_BIND_SERVICE` handelt, ist die Angabe des Ports oder von Portbereichen möglich.
- `-d`. Bei dem Objekt handelt es sich um eine Ausführungsdomäne (`EXEC DOMAIN`). Das Prinzip der Domäne wird in zukünftigen LIDS-Versionen abgeschafft!
- `-i level`. Vererbungstiefe des Zugriffes auf Kindprozesse des Subjektes. `-1` definiert eine unendliche Vererbung.
- `-t from-to`. Angabe des Zeitraumes, in dem die Regel gültig ist. Der Zeitraum von 09:00 Uhr bis 13:15 Uhr wird angegeben als `0900-1315`. Ein Datum kann nicht angegeben werden.
- `-j ACTION`. Dies erlaubt die Angabe des Zugriffes: `DENY`, `READONLY`, `APPEND`, `WRITE`, `IGNORE` oder `GRANT`. `GRANT` erlaubt die Zuweisung von Fähigkeiten (s.o.). Zusätzlich ist auch (ab Version 2.x.x) `DISABLE` möglich. Hiermit können Sie einzelne Funktionen abschalten (siehe Abschnitt »Wurmbekämpfung mit LIDS« auf S. 449). Dieses Target wird auch in Kombination mit Netfilter benötigt (siehe Abschnitt »LIDS und Netfilter« auf S. 448).

Diese Regeln können erzeugt (`lidsconf -A`) oder gelöscht (`lidsconf -D`) werden. Sie definieren, WER (`-s subject`) auf WAS (`-o object`) WANN (`-t from-to`) WIE (`-j ACTION`) zugreifen darf. Dabei kann zusätzlich ein Vererbungsgrad (`-i level`) definiert werden, der Kindprozessen gleichen Zugriff gewährt. Die Angabe von `-d` definiert eine `EXEC DOMAIN`. Zugriffe des Programms außerhalb des angegebenen Objektes sind nicht erlaubt. Dies wird weiter unten an einem Beispiel erläutert.

Beispiele

LIDS ist in der Lage, eine ähnliche Aufgabe wie Tripwire (siehe Abschnitt »Tripwire« auf S. 147) wahrzunehmen. Hierbei kann LIDS nicht nur den Zugriff auf Konfigurationsdateien und den Austausch von Systembefehlen protokollieren, sondern sogar unterbinden. Im Folgenden werden einige Beispiele aus der Praxis gegeben.

Die wichtigsten Systemadministrationsbefehle (wie `useradd`, `fdisk`, `fsck`, `iptables`, `route`, `ifconfig` etc.) befinden sich üblicherweise im Verzeichnis `/sbin`. Um nun jeden schreibenden, verändernden Zugriff auf dieses Verzeichnis zu unterbinden, kann die folgende Regel angewandt werden:

```
lidsconf -A -o /sbin -j READONLY
```

Diese Regel erlaubt nur noch lesenden Zugriff auf den Inhalt des Verzeichnisses.

Ein derartig eingeschränkter Zugriff wird häufig auch für das Verzeichnis */etc* gewünscht. Dies kann mit folgender Regel erreicht werden:

```
lidsconf -A -o /etc -j READONLY
```

Wurde LIDS nun aktiviert und versucht ein Benutzer einschließlich *root* eine Datei im Verzeichnis zu erzeugen, erhält er folgende Fehlermeldung:

```
# touch /etc/test
touch: creating `/etc/test': Operation not permitted
```

Diese Einschränkung erzeugt jedoch mehrere Probleme. Einige Dateien im Verzeichnis */etc* müssen häufig vom System modifiziert werden, andere verlangen strengere Einstellungen. Hierbei handelt es sich um:

- */etc/motd*. Diese Datei wird häufig neu erzeugt, um Benutzern bei ihrer Anmeldung Tagesinformationen (Message of the Day) zu übermitteln.
- */etc/mtab*. Diese Datei wird bei jedem Mountvorgang modifiziert.
- */etc/shadow*. Diese Datei sollte stärkeren Beschränkungen unterworfen werden, da sie die Kennwörter der Systembenutzer enthält. Hierzu zählen auch weitere Dateien, die sensible Daten enthalten wie zum Beispiel */etc/ppp/pap-secrets*, */etc/ppp/chap-secrets*, */etc/ipsec.secrets*, */etc/cipe/options* etc.

Um diesen Anforderungen gerecht zu werden, kann zunächst der Zugriff auf bestimmte Dateien erlaubt werden:

```
# Erlaube keinen schreibenden Zugriff auf /etc
lidsconf -A -o /etc -j READONLY
# Ausnahme für /etc/motd
lidsconf -A -o /etc/motd -j WRITE
# Ausnahme für den mount-Befehl und /etc/mtab
lidsconf -A -s /bin/mount -o /etc/mtab -j WRITE
lidsconf -A -s /bin/umount -o /etc/mtab -j WRITE
# Erlaube keinen Zugriff auf /etc/shadow
lidsconf -A -o /etc/shadow -j DENY
# Ausnahme login, su, sshd
lidsconf -A -s /bin/login -o /etc/shadow -j READONLY
lidsconf -A -s /bin/su -o /etc/shadow -j READONLY
lidsconf -A -s /usr/sbin/sshd -o /etc/shadow -j READONLY
```

Diese Regeln definieren genau den Zugriff auf das Verzeichnis */etc* und seine Dateien. Hierbei wird für die Datei */etc/shadow* eine Ausnahme angegeben. Dies erfolgt mit der Angabe des Subjektes, welches auf das angegebene Objekt zugreifen darf. Wer darf was? Die Befehle */bin/login*, */bin/su* und */usr/sbin/sshd* dürfen lesend auf die

Datei zugreifen. Ein schreibender Zugriff wird nicht erlaubt. Wenn die Benutzer ihre Kennwörter ändern möchten, so muss zusätzlich dem Befehl `/usr/bin/passwd` die Schreibberechtigung eingeräumt werden.

Da nun ein Angreifer durch Austausch des Befehls `/bin/su` Leseberechtigung an der Datei `/etc/shadow` erlangen würde, verlangt LIDS, dass Befehle, denen privilegierte Berechtigungen eingeräumt wurden, auch geschützt werden. Dies erfordert beim obigen Regelsatz zusätzlich die Regeln:

```
# Schütze Befehle
lidsconf -A -o /bin/login      -j READONLY
lidsconf -A -o /bin/su        -j READONLY
lidsconf -A -o /usr/sbin/sshd -j READONLY
```

LIDS speichert diese Regeln in einem eigenen Format in der Datei `/etc/lids/lids.conf` (s.o.). Diese Datei lässt sich leider nicht sehr gut verwalten. Daher empfiehlt es sich, die Datei nicht direkt zu editieren, sondern ein Script zu erzeugen, welches wiederum diese Datei erzeugt. Dazu werden einfach alle `lidsconf`-Aufrufe in einem Script zusammengefasst. Damit bei jedem Aufruf des Scripts der Regelaufbau wieder bei Null begonnen wird und nicht ältere Regeln zu Problemen führen, sollten zunächst alle Regeln gelöscht werden (`lidsconf -Z`). Ein fertiges Script sieht dann folgendermaßen aus:

```
# LIDS Script
# (c) 2002 Ralf Spenneberg
#
# Lösche alte Regeln
lidsconf -Z

# Schütze Befehle
lidsconf -A -o /bin/login      -j READONLY
lidsconf -A -o /bin/mount     -j READONLY
lidsconf -A -o /bin/umount    -j READONLY
lidsconf -A -o /bin/su        -j READONLY
lidsconf -A -o /usr/sbin/sshd -j READONLY

# Erlaube keinen schreibenen Zugriff auf /etc
lidsconf -A                -o /etc          -j READONLY
# Ausnahme für /etc/motd
lidsconf -A                -o /etc/motd    -j WRITE
# Ausnahme für den mount-Befehl und /etc/mtab
lidsconf -A -s /bin/mount    -o /etc/mtab -j WRITE
lidsconf -A -s /bin/umount   -o /etc/mtab -j WRITE
# Erlaube keinen Zugriff auf /etc/shadow
lidsconf -A -o /etc/shadow   -j DENY
# Ausnahme login, su, sshd
lidsconf -A -s /bin/login    -o /etc/shadow -j READONLY
lidsconf -A -s /bin/su       -o /etc/shadow -j READONLY
lidsconf -A -s /usr/sbin/sshd -o /etc/shadow -j READONLY
```

Bei jedem Aufruf des Scripts wird nun die LIDS-Regeldatenbank */etc/lids/lids.conf* neu erzeugt. Das Script selbst sollte ebenfalls besonders geschützt werden. Am einfachsten erfolgt dieser Schutz, wenn das Script im Verzeichnis */etc/lids* abgelegt wird. Dieses Verzeichnis ist automatisch beim Start von LIDS durch eine unsichtbare Regel geschützt.

Der Befehl `lidsconf` erlaubt auch die Definition von Zeitangaben und Fähigkeiten (Capabilities). Diese werden kurz erläutert, bevor allgemeine Grundregeln im nächsten Kapitel besprochen werden.

Protokolldateien (z.B. in */var/log*) stellen eine besondere Art von Datei unter Linux dar. Üblicherweise werden lediglich durch den Syslogd neue Meldungen an diese Dateien angehängt. Daher kann folgender Schutz für diese Dateien definiert werden:

```
lidsconf -A -o /var/log -j APPEND
```

Häufig wird jedoch zusätzlich eine Rotation der Protokolle gewünscht. Damit dies möglich ist, muss der Befehl zur Rotation Schreibrechte besitzen. Dadurch wird jedoch ein Sicherheitsproblem erzeugt. Ein Angreifer könnte diesen Zusammenhang erkennen und durch manuellen Start des Befehls `logrotate` (oder ähnlich) die Protokolle so lange rotieren, bis sie gelöscht werden. Meist werden die Protokolle nach einigen Rotationsdurchläufen gelöscht.

In diesem Fall ist es besser, dem `cron`-Dämon die entsprechenden Rechte zu geben, so dass er sie mittels Vererbung auf den `logrotate`-Befehl überträgt. Ein Angreifer könnte auch hier eine mögliche Sicherheitslücke im `cron`-Dämon ausnutzen und die Protokolle löschen. Zusätzlich sollte daher die Rotation nur in dem Zeitraum erlaubt werden, in dem sie auch tatsächlich durchgeführt wird. Unter Red Hat Linux ist dies der Zeitraum von 4:02 bis 4:04 morgens.

```
lidsconf -A -o /var/log -j APPEND
lidsconf -A -o /usr/sbin/crond -j READONLY
lidsconf -A -o /etc/crontab -j DENY
lidsconf -A -s /usr/sbin/crond -o /etc/crontab -j READONLY
lidsconf -A -s /usr/sbin/crond -o /var/log -i 2 -t 0402-0404 -j WRITE
```

Die Datei */etc/lids/lids.cap* enthält die Fähigkeiten, die nach der Versiegelung des Kernels für `root` verbleiben. Benötigt ein Prozess Fähigkeiten, die über diese hinausgehen, so müssen sie ihm spezifisch zugewiesen werden. So benötigt der Webserver *Apache* zum Beispiel das Recht, auf den Port 80 zuzugreifen. Dies ist die Fähigkeit `CAP_NET_BIND_SERVICE`. Entweder wird der Webserver *Apache* gestartet, bevor der Kernel versiegelt wurde (und diese Fähigkeit entfernt wurde) oder die Fähigkeit muss dem Prozess spezifisch zugewiesen werden:

```
lidsconf -A -o /usr/sbin/httpd -j READONLY
lidsconf -A -s /usr/sbin/httpd -o CAP_NET_BIND_SERVICE 80,443 -j GRANT
```

Außerdem benötigt der Webserver die Fähigkeit, seinen Benutzerkontext von *root* zu einem anderen Benutzer (*apache*) zu schalten. Dies sind die Fähigkeiten `CAP_SETUID` und `CAP_SETGID`. Dies wird weiter unten bei den Fallbeispielen erneut aufgegriffen.

Aktualisierung der Regeln

Modifikationen der Datenbank `/etc/lids/lids.conf` mit dem Werkzeug `lidsconf` oder mit einem Editor haben keine direkte Auswirkung auf das System. Diese Änderungen sind erst nach einem Neustart des Systems und erneuter Versiegelung des Kernels aktiv. Auch ein Neustart des LIDS-Systems im Kernel wird diese Veränderungen nicht bemerken und aktivieren.

Wurde jedoch LIDS im Kernel mit der Option `CONFIG_LIDS_RELOAD_CONF` übersetzt, so besteht die Möglichkeit, die Konfiguration mit dem Befehl

```
lidsadm -S -- +RELOAD_CONF
```

neu zu laden. Ein weitere Aktualisierung der Datenbank ist erforderlich, wenn die eigentlich zu überwachenden Dateien ausgetauscht wurden. Der Befehl `passwd` wird bei jeder Modifikation eines Kennwortes die Datei `/etc/shadow` neu erzeugen. Diese neue Datei wird eine andere Inode-Nummer aufweisen. Da die Datenbank `/etc/lids/lids.conf` und der Kernel die Sicherheitseinstellungen über die Inode-Nummer definieren, müssen diese Strukturen aktualisiert werden. Das erfolgt mit dem Befehl:

```
lidsconf -U
```

11.6.5 Grundregeln für jeden Rechner

LIDS ist in der Lage, eine vollkommene Überwachung des Rechners durchzuführen. Die Konfiguration von LIDS ist jedoch sehr aufwändig und die spätere Administration schwierig. Daher sollte im Vorfeld genau abgewogen werden, welche Rechner mit LIDS gesichert werden sollen und für welche Dateien LIDS verantwortlich sein soll. In vielen Fällen kann mit dem Lern-Modus (siehe Abschnitt »LIDS Learning-Mode/ACL-Discovery« auf S. 450) die Arbeit stark vereinfacht werden. Rechner, auf denen viele Benutzer angelegt wurden, die wöchentlich ihre Kennwörter ändern, sind mit LIDS schwer zu überwachen, da LIDS den Zugriff auf die Dateien über die Inode-Nummer kontrolliert. Hier ist zu überlegen, ob zentrale Authentifizierungsserver (LDAP, Kerberos, NIS) eingesetzt werden können.

Im Folgenden werden die wichtigsten Regeln, welche im Grunde auf allen Rechnern eingesetzt werden können, vorgestellt. Weitere Anregungen finden sich auf der LIDS-Homepage <http://www.lids.org> und dem LIDS-FAQ, welches über die Homepage erreichbar ist.

Zunächst sollte ein Schutz der binären, ausführbaren Programme erfolgen. Wenn sämtliche Installationen auf dem System abgeschlossen wurden, sind keine schreibenden Zugriffe auf das System mehr nötig.

```

# LIDS Script
# (c) 2002 Ralf Spenneberg
#
# Lösche alte Regeln
lidsconf -Z

# Binärdateien und Bibliotheken
lidsconf -A                -o /sbin                -j READONLY
lidsconf -A                -o /bin                  -j READONLY
lidsconf -A                -o /lib                  -j READONLY
lidsconf -A                -o /usr                  -j READONLY

# Erforderlich wenn sich /usr/local auf einer eigenen Partition befindet
lidsconf -A                -o /usr/local            -j READONLY

# Systemkonfigurationsdateien
lidsconf -A                -o /etc                  -j READONLY
lidsconf -A                -o /etc/shadow           -j DENY

# Abhängig von dem Bootmanager
lidsconf -A                -o /boot/grub/grub.conf -j DENY
#lidsconf -A                -o /etc/lilo.conf        -j DENY

# Erlaube Anmeldungen
lidsconf -A -s /bin/login   -o /etc/shadow   -j READONLY
lidsconf -A -s /usr/sbin/sshd -o /etc/shadow   -j READONLY
lidsconf -A -s /bin/su      -o /etc/shadow   -j READONLY
lidsconf -A -s /bin/su      -o CAP_SETUID    -j READONLY
lidsconf -A -s /bin/su      -o CAP_SETGID    -j READONLY

# Protokolldateien
lidsconf -A                -o /var/log             -j APPEND
lidsconf -A -s /bin/login   -o /var/log/wtmp  -j WRITE
lidsconf -A -s /bin/login   -o /var/log/lastlog -j WRITE
lidsconf -A -s /usr/sbin/sshd -o /var/log/wtmp  -j WRITE
lidsconf -A -s /usr/sbin/sshd -o /var/log/lastlog -j WRITE

# Herunterfahren ermöglichen
lidsconf -A -s /etc/rc.d/init.d/halt -o CAP_KILL      -i 1 -j GRANT
lidsconf -A -s /etc/rc.d/init.d/halt -o CAP_NET_ADMIN -i 1 -j GRANT
lidsconf -A -s /etc/rc.d/init.d/halt -o CAP_SYS_ADMIN -i 1 -j GRANT

```

Die letzten Regeln im angegebenen Script erlauben es diesem, den Rechner herunterzufahren, die Dienste zu beenden und eingebundene Dateisysteme auszuhängen.

11.7 Welche Rechner sollen mit LIDS geschützt werden?

Wie bereits im letzten Kapitel ausgeführt wurde, macht es keinen Sinn, jeden Rechner von LIDS überwachen zu lassen. Der administrative Aufwand wird dadurch enorm gesteigert. Viele Funktionen von LIDS können auch durch sinnvollen Einsatz von Tripwire zur Verfügung gestellt werden. LIDS kann aber ideal in einer Art Hybrid-System mit Tripwire zum Einsatz kommen. Hierbei kann LIDS den Zugriff auf die Tripwire-Dateien überwachen und diese vor Modifikationen schützen. Im Folgenden werden reine LIDS-basierte Konfigurationen für die entsprechenden Dienste vorgestellt. Diese Konfigurationsvorschläge stellen nur Beispiele dar und benötigen in vielen Umgebungen weitere Anpassungen.

11.7.1 Webserver

Die folgende Konfiguration definiert die Regeln für einen *Apache*-Webserver. Hierbei gehen die Regeln von den Standardpfaden einer Red Hat Linux-Distribution aus. Das bedeutet, das Protokollverzeichnis ist */var/log/httpd*, das Konfigurationsverzeichnis ist */etc/httpd* und die Webseiten befinden sich in */var/www*. Insbesondere die privaten Schlüssel eines SSL-Webservers sollten geschützt werden und befinden sich meist im Konfigurationsverzeichnis. Wenn ein Angreifer den privaten Schlüssel lesen kann und dieser nicht mit einer Passphrase geschützt wurde, besteht die Möglichkeit, einen Man-in-the-Middle-Angriff auf die SSL-Verschlüsselung zu starten. Es ergeben sich anschließend folgende Regeln:

```
# Schütze einen Webserver

# Binärdatei
lidsconf -A                -o /usr/sbin/httpd  -j READONLY

# Module
lidsconf -A                -o /usr/lib/apache  -j READONLY

# Konfiguration (speziell auch der private SSL Schlüssel)
lidsconf -A                -o /etc/httpd      -j DENY
lidsconf -A -s /usr/sbin/httpd -o /etc/httpd      -j READONLY

# Webseiten
lidsconf -A                -o /var/www          -j DENY
lidsconf -A -s /usr/sbin/httpd -o /var/www          -j READONLY

# Protokolle
lidsconf -A                -o /var/log/httpd    -j DENY
lidsconf -A -s /usr/sbin/httpd -o /var/log/httpd    -j WRITE

# Fähigkeiten
lidsconf -A -s /usr/sbin/httpd -o CAP_SETUID      -j GRANT
```

```

lidsconf -A -s /usr/sbin/httpd -o CAP_SETGID -j GRANT
# Wenn Apache nicht zu Beginn gestartet wird
lidsconf -A -s /usr/sbin/httpd -o CAP_NET_BIND_SERVICES 80,443 -j GRANT

```

11.7.2 DNS-Server

Die folgende Konfiguration definiert die Regeln für einen *Bind 9* DNS-Server. Hierbei gehen die Regeln von den Standardpfaden einer Red Hat Linux-Distribution aus. Das bedeutet, die Konfigurationsdateien sind */etc/named.conf*, */etc/rndc.conf* und */etc/rndc.key*. Die Zonendateien befinden sich in */var/named*. Insbesondere der Schlüssel zur Administration und die Schlüssel zur Signatur der Zonendateien sollen geschützt werden. Wenn ein Angreifer diese privaten Schlüssel lesen kann, besteht die Möglichkeit einer Kompromittierung des DNS-Servers. Daraus ergeben sich anschließend folgende Regeln:

```

# Schütze einen Bind DNS-Server

# Binärdateien und Bibliotheken
lidsconf -A -o /usr/ -j READONLY

# Konfiguration (speziell auch der private Schlüssel)
lidsconf -A -o /etc/named.conf -j DENY
lidsconf -A -o /etc/rndc.conf -j DENY
lidsconf -A -o /etc/rndc.key -j DENY
lidsconf -A -s /usr/sbin/named -o /etc/named.conf -j READONLY
lidsconf -A -s /usr/sbin/rndc -o /etc/rndc.key -j READONLY
lidsconf -A -s /usr/sbin/rndc -o /etc/rndc.conf -j READONLY

# Zonen
lidsconf -A -o /var/named -j DENY
lidsconf -A -s /usr/sbin/named -o /var/named -j READONLY

# Fähigkeiten
lidsconf -A -s /usr/sbin/named -o CAP_SETUID -j GRANT
lidsconf -A -s /usr/sbin/named -o CAP_SETGID -j GRANT
lidsconf -A -s /usr/sbin/named -o CAP_SETPCAP -j GRANT
lidsconf -A -s /usr/sbin/named -o CAP_SYS_CHROOT -j GRANT
lidsconf -A -s /usr/sbin/named -o CAP_SYS_RESOURCE -j GRANT
# Wenn Bind nicht zu Beginn gestartet wird
lidsconf -A -s /usr/sbin/named -o CAP_NET_BIND_SERVICES 53 -j GRANT

```

11.7.3 Proxy-Server

Die folgende Konfiguration definiert die Regeln für einen *Squid* Proxy-Server. Hierbei gehen die Regeln von den Standardpfaden einer Red Hat Linux-Distribution aus.

Das bedeutet, das Protokollverzeichnis ist */var/log/squid*, das Konfigurationsverzeichnis ist */etc/squid* und der Cache befindet sich in */var/spool/squid*. Insbesondere die Benutzerdatenbank von Squid mit den verschlüsselten Kennwörtern sollte geschützt werden. Sie befindet sich meist im Konfigurationsverzeichnis. Daraus ergeben sich anschließend folgende Regeln:

```
# Schütze einen Squid Proxy-Server

# Binärdateien und Hilfsprogramme
lidsconf -A                -o /usr/sbin/squid           -j READONLY
lidsconf -A                -o /usr/sbin/client          -j READONLY
lidsconf -A                -o /usr/lib/squid           -j READONLY

# Konfiguration (speziell auch die Benutzerdatenbank)
lidsconf -A                -o /etc/squid                -j DENY
lidsconf -A -s /usr/sbin/squid -o /etc/squid          -i 2 -j READONLY

# Cache
lidsconf -A                -o /var/spool/squid          -j DENY
lidsconf -A -s /usr/sbin/squid -o /var/spool/squid      -i 2 -j WRITE

# Protokolle
lidsconf -A                -o /var/log/squid            -j DENY
lidsconf -A -s /usr/sbin/squid -o /var/log/squid        -i 2 -j WRITE

# Fähigkeiten (Wenn Squid sich auf einen privilegierte Port binden soll)
lidsconf -A -s /usr/sbin/squid -o CAP_NET_BIND_SERVICES 80 -j GRANT
```

11.7.4 Firewall/Snort-Sensor

Die folgende Konfiguration definiert die Regeln für einen kombinierten *Firewall/Snort*-Rechner. Hierbei gehen die Regeln von den Standardpfaden einer Red Hat Linux-Distribution aus. Das bedeutet, die Protokolle befinden sich in */var/log/* und */var/log/snort*. Die Konfiguration befindet sich in */etc/snort* und */etc/sysconfig/iptables*. Insbesondere die Regeln der Firewall und des Snort-Sensors sollten geschützt werden. Daraus ergeben sich anschließend folgende Regeln:

```
# Schütze eine Firewall

# Binärdateien und Hilfsprogramme
lidsconf -A                -o /usr/sbin/snort           -j READONLY
lidsconf -A                -o /sbin/iptables           -j READONLY
lidsconf -A                -o /sbin/iptables-save       -j READONLY
lidsconf -A                -o /sbin/iptables-restore    -j READONLY
lidsconf -A                -o /lib/iptables             -j READONLY
lidsconf -A                -o /etc/rc.d/init.d/iptables -j READONLY
```

```

# Konfiguration
lidsconf -A -o /etc/snort -j DENY
lidsconf -A -s /usr/sbin/snort -o /etc/snort -j READONLY
lidsconf -A -o /etc/sysconfig/iptables -j DENY
lidsconf -A -s /sbin/iptables-restore -o /etc/sysconfig/iptables -j READONLY
lidsconf -A -s /etc/rc.d/init.d/iptables -o /etc/sysconfig/iptables -j READONLY

# Protokolle
lidsconf -A -o /var/log/snort -j DENY
lidsconf -A -s /usr/sbin/snort -o /var/log/snort -j WRITE
lidsconf -A -s /usr/sbin/snort -o /var/log/secure -j APPEND
lidsconf -A -o /var/log/messages -j APPEND

# Fähigkeiten
lidsconf -A -s /usr/sbin/snort -o CAP_DAC_OVERRIDE -j GRANT
lidsconf -A -s /usr/sbin/snort -o CAP_NET_RAW -j GRANT
lidsconf -A -s /usr/sbin/snort -o CAP_SETUID -j GRANT
lidsconf -A -s /usr/sbin/snort -o CAP_SETGID -j GRANT
lidsconf -A -s /usr/sbin/snort -o CAP_CHROOT -j GRANT

# (Wenn Snort unsichtbar sein soll)
lidsconf -A -s /usr/sbin/snort -o CAP_HIDDEN -j GRANT

```

11.8 LIDS-Protokollierung durch LIDS

LIDS kennt zwei verschiedene Methoden der Protokollierung: Eintrag in der Protokolldatei und E-Mail. Ein möglicher Eintrag in der Protokolldatei */var/log/messages* sieht wie folgt aus:

```

Jul  6 17:56:42 localhost kernel: LIDS: dhcpcd (dev 8:2 inode 402845) pid 463
↳ ppid 426 uid/gid (0/0) on (null tty) : Attempt to mkdir /etc/dhcpc
Jul  6 17:56:42 localhost kernel: LIDS: rpc.statd (dev 8:2 inode 404463) pid
↳ 538 ppid 1 uid/gid (29/29) on (null tty) : violated CAP_NET_BIND_SERVICE

```

Diese Protokolleinträge sind selbsterklärend. Sie definieren genau, welcher Befehl mit welcher Prozessnummer versucht hat, eine nicht erlaubte Aktion durchzuführen. Wenn Sie beginnen, LIDS einzusetzen und zu konfigurieren, werden Sie viele derartige Einträge in Ihren Protokollen sehen. Diese helfen Ihnen bei der Fehlersuche.

Die zweite Möglichkeit der Protokollierung ist E-Mail. LIDS besitzt einen eigenen E-Mail-Client im Kernel. Wurde dieser bei der Kernel-Konfiguration aktiviert, so erfolgt die Konfiguration mit der Datei */etc/lids/lids.net*. Wurde die Konfiguration der Datei nicht verändert (siehe Abschnitt »LIDS 1.x.x Konfigurationsdateien« auf S. 433, »Einführung«), dann versendet LIDS folgende E-Mails:

Message 1:

From LIDS_ALERT@lids.sinocluster.com Wed Jun 19 12:01:41 2002
 Date: Wed, 19 Jun 2002 12:01:41 +0200
 From: LIDS_ALERT@lids.sinocluster.com
 To: root@localhost.localdomain
 Subject: LIDS ALert

LIDS: dhcpcd (dev 8:2 inode 402845) pid 483 ppid 446 uid/gid (0/0) on (null
 ↳ tty) : Attempt to mkdir /etc/dhcpc

Message 2:

From LIDS_ALERT@lids.sinocluster.com Wed Jun 19 12:01:41 2002
 Date: Wed, 19 Jun 2002 12:01:41 +0200
 From: LIDS_ALERT@lids.sinocluster.com
 To: root@localhost.localdomain
 Subject: LIDS ALert

LIDS: rpc.statd (dev 8:2 inode 404463) pid 558 ppid 1 uid/gid (29/29) on
 ↳ (null tty) : violated CAP_NET_BIND_SERVICE

11.9 LIDS und Netfilter

Die Linux-Paketfilter-Maschine *Netfilter* kümmert sich um die Firewallfunktionalität des Linux-Kernel 2.4 und 2.6. Sie erlaubt die Markierung von Paketen und die Filterung der Pakete in Abhängigkeit von der Markierung. LIDS kann in der Version 2.x.x für den Linux-Kernel 2.6 alle Pakete eines bestimmten Prozesses mit dem Netfilter Mark Target markieren. Hierzu müssen bei der Übersetzung des Kernels die entsprechenden Netfilter und Module (insbesondere *ipt_MARK* und *ipt_mark* aktiviert werden. Bei der LIDS-Konfiguration müssen Sie darauf achten, dass der *LIDS NETFILTER Network Mark* aktiviert wird.

Für die Konfiguration zur Laufzeit sollten Sie darauf achten, dass die entsprechenden Module und Regeln geladen werden:

```
# modprobe ip_tables
# modprobe iptable_mangle
# modprobe ipt_MARK
# modprobe ipt_mark
# iptables -F -t mangle
# iptables -A OUTPUT -t mangle -j MARK --set-mark 0
```

Die letzte Regel ist als Dummy-Regel erforderlich, da ansonsten auch die LIDS-Markierung nicht durchgeführt wird.

Nun können Sie die LIDS-Konfiguration anpassen, so dass zum Beispiel alle Pakete des Secure Shell-Clients markiert werden:

```
# lidsconf -A -s /usr/sbin/ssh -o LIDS_SOCKET_NF_MARK 5 -j DISABLE
```

Lassen Sie sich von dem `DISABLE` nicht verwirren. Hier führt diese Regel zur Markierung der Pakete.

Nun können Sie Regeln erzeugen, die die Markierung auswerten.

```
# iptables OUTPUT -m mark --mark 5 ! -d www.spenneberg.org -j REJECT
```

Diese Regel verwirft alle SSH-Client-Pakete, die nicht an *www.spenneberg.org* gerichtet sind.

11.10 Wurmbekämpfung mit LIDS

LIDS verfügt über neue Funktionen, die es ermöglichen, mit LIDS neue Würmer zu unterbinden. Ein Wurm ist ein Programm, welches sich selbst kopiert und verbreitet. Es braucht im Gegensatz zum Virus nicht eine Benutzerinteraktion oder ein Wirtsprogramm. Seit dem ersten Wurm (Morris-Wurm, siehe <http://de.wikipedia.org/wiki/Morris-Wurm>) bis hin zu den modernen Würmern (z.B. Sasser) haben die Würmer an Schadenspotential zugelegt. Sie waren und sind in der Lage, große Teile des Internets lahm zu legen.

LIDS kann zur Bekämpfung der Wurmverbreitung durch den eigenen Rechner eingesetzt werden. So befallen viele Würmer einen Rechner über eine Sicherheitslücke eines Netzwerkdienstes. Dies erfolgt üblicherweise durch einen Bufferoverflow bei dem Code injiziert wird. Dieser Code lädt anschließend den Wurm herunter, kompiliert und startet ihn. Mit LIDS können Sie dem Apache-Webserver das Privileg, Verbindungen zu öffnen, nehmen. Nach einem erfolgreichen Einbruch auf dem Apache-Webserver kann der Wurm dann nicht heruntergeladen werden. Hierfür wird das `DISABLE`-Target des `lidsconf`-Befehls verwendet:

```
# lidsconf -A -o /usr/sbin/httpd -j READ
# lidsconf -A -s /usr/sbin/httpd -o LIDS_SOCKET_CREATE -j DISABLE
```

Nun kann der `httpd`-Prozess keine Netzwerkverbindung mehr aufbauen.



Achtung:

Auch Verbindungen mit DNS-Server und SQL-Datenbanken werden nun wirkungsvoll unterbunden. Falls Ihre Webseite die Inhalte dynamisch mit PHP aus Datenbanken generiert, können Sie diese Funktion nicht nutzen. Dann können Sie aber alle Pakete des Web-servers mit LIDS markieren und spezifisch nur die Verbindung zu den bekannten DNS-Servern und Datenbanken erlauben.

11.11 LIDS Learning-Mode/ACL-Discovery

Learning-Mode ist eine neue Funktion, die Mitte 2003 Einzug in LIDS gehalten hat. Dieser Modus wurde inzwischen in ACL-Discovery umbenannt. Er steht im Moment nur ab der Version 2.x für den Linux-Kernel 2.6 zur Verfügung. Während Sie in der Vergangenheit mühsam alle ACLs selbst mit Trial and Error ermitteln mussten, kann nun LIDS Sie dabei unterstützen.

Im Lern-Modus blockiert LIDS keinen Prozess, sondern protokolliert lediglich die verschiedenen Zugriffe. Zusätzlich erzeugt LIDS auch entsprechende Regeln, die anschließend genau dieses Verhalten erlauben würden.

Die Aktivierung des Lern-Modus kann auf zwei verschiedene Arten erfolgen:

- Sie können in der Datei `/etc/lids/lids.ini` den Parameter `ACL_DISCOVERY=1` setzen. Beim nächsten Start von LIDS wird der Lern-Modus aktiviert.
- Wenn der Rechner sich bereits im LIDS-geschützten Modus befindet, können Sie mit dem Befehl `lidsadm -S -- +ACL DISCOVERY` den Lern-Modus anschalten.

Um den Lern-Modus abzuschalten, setzen Sie in der Datei `/etc/lids/lids.ini` den Parameter `ACL_DISCOVERY=0`.

Im Lern-Modus wird LIDS auch alle Prozesse überwachen. Jedoch werden Übertretungen nicht verhindert, sondern lediglich protokolliert. Zusätzlich gibt LIDS eine Regel aus, die diese Übertretung erlaubt hätte:

```
LIDS: login (dev 1:0 inode 783732) pid 211 ppid 1 uid/gid (0/0) on (tty1) :
↳ atte
mpt to open lastlog for writing - logging disabled for 60s
LIDS_ACL_DISCOVERY:[state 1]783732:770:login:7:0:16066:773:lastlog:0-0
```

Das Vorgehen im Lern-Modus erfolgt üblicherweise in den folgenden Schritten:

1. Booten Sie einen Kernel mit LIDS-Unterstützung und aktivieren Sie den Lern-Modus in der Datei `lids.ini`.
2. Versiegeln Sie das System: `lidsadm -I`.
3. Führen Sie die normalen Operationen durch.
4. Wechseln Sie in den Shutdown-Zustand: `lidsadm -S -- +SHUTDOWN`.
5. Booten Sie das System neu mit einem Kernel ohne LIDS-Unterstützung.
6. Rufen Sie das Programm `lids_acl_discovery.pl` auf. Dieses Programm finden Sie in `lids/tools/acl_discovery`. Dieser Befehl generiert drei ACL-Dateien: `lids.boot.conf`, `lids.postboot.conf` und `lids.shutdown.conf`.

Überprüfen Sie die erzeugten Dateien und nehmen Sie eventuelle Änderungen vor. Wichtig ist vor allem, dass Sie generische Befehle und deren Rechte überprüfen. Wenn die Bash über Schreibrechte in sensitiven Verzeichnissen verfügt, ist das nicht besonders sicher!

11.12 LIDS Trusted Path Execution

LIDS unterstützt ab März 2004 die so genannte Trusted Path Execution (TPE). Hierbei handelt es sich um die Eigenschaft, nur die Programme auszuführen, deren Integrität sichergestellt werden kann.

Trusted Path-Funktionalität wird auch von verschiedenen anderen Produkten sichergestellt. Hierbei werden üblicherweise alle Benutzer in zwei Gruppen (*trusted* und *untrusted*) eingeteilt. Genauso werden alle Programme automatisch in diese Gruppen eingeteilt: Alle Programme, die sich in einem vertrauenswürdigen Pfad befinden, sind *trusted*. Hierzu analysieren die Systeme den Pfad und prüfen, ob andere Personen als *root* den Pfad oder den Befehl modifizieren können. Ist dies der Fall, so wird das Programm als *untrusted* angesehen.

Inspiziert von diesen Verfahren hat Yusuf Wilatjati Purna im März 2004 ein ähnliches Verfahren in LIDS implementiert. Diese geht jedoch über die meisten üblichen Implementierungen hinaus, da sie auch die Bibliotheken und Kernel-Module mit einbezieht. So werden alle Programme, Bibliotheken und Kernel-Module analysiert und als *trusted* oder *untrusted* eingeordnet. Dies hängt davon ab, ob ein Benutzer (auch *root*) diese Dateien oder ihren Pfad modifizieren kann. Sobald ein Verzeichnis nicht von LIDS als *READONLY* überwacht wird, werden die enthaltenen Dateien als *untrusted* betrachtet.

Um nun nur *trusted* Programme, Bibliotheken und Kernel-Module zu erlauben, muss lediglich der TPE-Modus aktiviert werden. Zusätzliche Regeln sind nicht erforderlich.

Die Aktivierung erfolgt mit:

```
# lidsadm -S -- +TPE
```

Genauso können Sie den Modus auch wieder deaktivieren:

```
# lidsadm -s -- -TPE
```

Sie können den Modus auch direkt bei der Versiegelung des Kernels aktivieren:

```
# lidsadm -I +TPE
```



Achtung:

Der Entwickler weist selbst darauf hin, dass die Implementierung noch nicht perfekt ist. So ist der TPE-Modus noch nicht in der Lage, durch Shells ausgeführte Skripte zu prüfen. Außerdem kann der TPE-Modus nicht die Ausführung von injiziertem Code durch zum Beispiel Bufferoverflows verhindern.

11.13 LIDS Trusted Domain Enforcement

Speziell das Problem von Bufferoverflows in privilegierten Applikationen soll das Trusted Domain Enforcement (TDE) lösen. LIDS kann normalerweise nicht erkennen, dass eine vertraute Applikation sich plötzlich nicht mehr wie erwartet verhält und von einem Angreifer ausgenutzt wird.

TDE implementiert eine TDE-Richtlinie und führt Applikationen in einer Art Sandkasten (Sandbox) aus.

11.13.1 TDE-Richtlinie

Sobald TDE im Kernel konfiguriert wurde, aktiviert der Kernel die folgende Richtlinie:

Ein privilegierter Prozess, der ungeschützte Eingaben entgegennimmt, verliert alle Privilegien.

Die einzige Möglichkeit, dies für bestimmte Prozesse, die ungeschützte Daten entgegennehmen müssen, zu deaktivieren, ist die Zuweisung der neuen Capability `CAP_PROTECTED`.

11.13.2 TDE-Sandbox

Bei der Anwendung der Sandbox müssen Sie das spätere Verhalten der Applikation exakt mit ACLs beschreiben. Das bedeutet, Sie müssen für jede Datei, auf die die Applikation zugreifen möchte, eine ACL erzeugen, die den Zugriff erlaubt. Jeder nicht spezifisch erlaubte Zugriff ist automatisch verboten.

Sie können dieses Verhalten für eine bestimmte Applikation anschalten. Hierzu rufen Sie folgenden Befehl auf:

```
# lidsconf -A -s /opt/mybash -o LIDS_SANDBOX -j ENABLE
```

Ein wesentlich kompletteres Beispiel für Apache und Samba ist in dem Dokument *LIDS-TDE-feature.txt* beschrieben, welches im Dokumentationsbereich der Homepage gefunden werden kann.

11.14 LIDS-Zusammenfassung

LIDS ist ein kernelbasiertes IDS-System, welches die Zugriffsrechte des Linux-Systems neu definiert. Linux- und UNIX-Systeme besitzen üblicherweise einen allmächtigen Benutzer *root*. Die Rechte dieses Benutzers können mit LIDS entzogen und einzelnen Prozessen zugewiesen werden. Somit kann ein Einbrecher, auch wenn

er Zugriff auf das *root*-Konto des Systems erlangt hat, kaum Änderungen am System vornehmen.

Die Konfiguration und Administration von LIDS ist recht umständlich und mühsam. Daher ist es häufig sinnvoll, LIDS in Zusammenarbeit mit anderen Systemen wie zum Beispiel Tripwire einzusetzen.

