

Ralf Spenneberg

# Linux-Firewalls mit iptables & Co.

Sicherheit mit Kernel 2.4 und 2.6  
für Linux-Server und -Netzwerke



 ADDISON-WESLEY

---

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England  
Don Mills, Ontario • Sydney • Mexico City  
Madrid • Amsterdam



# 24 Fortgeschrittene Protokollierung

Wenn Sie die Protokolle Ihrer Firewall anspruchsvoll weiterverarbeiten möchten, werden Sie möglicherweise schnell an die Grenzen des LOG-Targets stoßen. Dann sollten Sie sich das ULOG-Target ein wenig genauer ansehen. Es gibt mehrere Softwarepakete, die mit diesem Target zusammenarbeiten und die Protokolle schreiben können. Zwei davon möchte ich Ihnen vorstellen: `ulogd` und `specter`.

Die Auswertung kann mit einigen der im Kapitel 12, Protokollanalyse, vorgestellten Werkzeuge erfolgen.

## 24.1 Das ULOG-Target

Dieses Ziel erlaubt es Ihnen, Pakete zur Protokollierung an eine beliebige Userspace-Applikation weiterzuleiten. Diese Applikation muss den `netlink`-Socket betrachten. Am einfachsten verwenden Sie `ulogd` oder `specter` für die Protokollierung. Diese nehmen die Parameter und Pakete entgegen und können die Protokollierung in einer Datei oder einer Datenbank vornehmen (siehe Kapitel 24).

Damit der Protokolldienst zwischen den verschiedenen Protokolldateien oder Datenbanken unterscheiden kann, können Sie die Protokollierung mit den folgenden Optionen genauer bestimmen:

- `--ulog-nlgroup <nlgroup>`: ULOG unterstützt 32 verschiedene Netlink-Gruppen (1-32). Sie können in dem `ulogd` für jede Gruppe zum Beispiel eine andere Datenbank definieren.
- `--ulog-prefix »Zeichenkette«`: Ähnlich dem LOG-Target können Sie auch hier eine Zeichenkette (maximal 32 Zeichen) definieren, die jeder Meldung vorangestellt wird.
- `--ulog-cprange <paketgröße>`: Für die spätere Analyse ist es nützlich, das Paket oder zumindest Teile des Pakets untersuchen zu können. Hiermit können Sie angeben, wie viele Bytes des Pakets an das Userspace-Programm übergeben und damit möglicherweise protokolliert werden sollen. Der Default-Wert 0 kopiert das ganze Paket.
- `--ulog-qthreshold <queuegröße>`: Da die Übergabe des Pakets vom dem Kernel in den Userspace mit aufwändigen Überprüfungen verbunden ist, können Sie hier de-

finieren, wie viele Pakete der Kernel vor der Übergabe sammeln (1-50) und gemeinsam übergeben soll. Der Default-Wert ist aus Gründen der Rückwärtskompatibilität 1. Diesen Wert können Sie für jede Regel einzeln definieren.

```
iptables -A FORWARD -p icmp -m length --length 200: -j ULOG \
--ulog-nlgroup 2 --ulog-prefix "Großes ICMP: " \
--ulog-qthreshold 10
```

## 24.2 Das ipt\_ULOG-Kernelmodul

Beim Laden des ipt\_ULOG-Kernelmoduls können Sie einige zusätzliche Parameter definieren, die das Verhalten und die Geschwindigkeit beeinflussen können. Diese Optionen möchte ich Ihnen hier vorstellen. Wenn nicht alle hier aufgeführten Optionen von Ihrem Modul unterstützt werden, setzen Sie wahrscheinlich eine ältere Kernel-Version ein. Um diese Optionen zu nutzen, müssen Sie vor der ersten Regel, die das ULOG-Target verwendet, das Modul manuell mit `modprobe` laden.

- `nbufsiz=<zahl>`: Hiermit geben Sie die Größe des Netlink-Puffers an. Der Puffer darf maximal eine Größe von 128 KByte haben. Größere Puffer erhöhen die Geschwindigkeit, verzögern aber möglicherweise die Protokollierung leicht. Default: 4096.
- `flushtimeout:int`: Dieser Wert definiert, in welchen Intervallen spätestens der Puffer geleert werden muss, auch wenn der Puffer nicht voll ist. Damit kann die mögliche Verzögerung durch einen großen Puffer beschränkt werden. Die Zeiteinheit ist 10 ms auf x86-Systemen. Der Default ist 10 (100 ms).
- `nflog:int`: Diese Option definiert, ob dieses Modul auch für die Filterung von internen Netfilter-Meldungen verwendet werden soll. Dies sind zum Beispiel Meldungen über ungültige Pakete. Der Default ist 1 (Ja).

## 24.3 Der Ulogd-Daemon

Der Ulogd-Daemon wurde von Harald Welte geschrieben und existiert im Moment in zwei Versionen. Auf <http://gnumonks.org/projects> finden Sie den Ulogd-Daemon in der Version 1 für die Verwendung mit dem ULOG-Target. Wenn Sie bereits den Kernel 2.6.14 mit dem NFLOG-Target einsetzen, können Sie den Ulogd-Daemon in der Version 2 verwenden, der im Moment (Okt. 2005) nur auf dem Subversion-Server verfügbar ist <http://svn.gnumonks.org/branches/ulog/ulogd2/>. Um den Ulogd-2-Daemon auszuchecken und zu übersetzen, verwenden Sie:

```
$ svn co http://svn.gnumonks.org/branches/ulog/ulogd2
$ cd ulogd2/
$ aclocal
$ automake
$ autoconf
$ ./configure
$ make
```

Der Ulogd verwendet Plugins, um die verschiedenen Funktionen zur Verfügung zu stellen. Diese Plugins werden in einer Konfigurationsdatei beschrieben, in der Sie festlegen, welche Meldungen wie protokolliert werden sollen. Eine Beispieldatei ist im Folgenden abgedruckt:

```
# Example configuration for ulogd
# $Id: ulogd.conf.in 714 2005-02-19 21:33:43Z laforge $
#

[global]
#####
# GLOBAL OPTIONS
#####

# netlink multicast group (the same as the iptables --ulog-nlgroup param)
nlgroup=1

# logfile for status messages
logfile="/var/log/ulogd/ulogd.log"

# loglevel: debug(1), info(3), notice(5), error(7) or fatal(8)
loglevel=5

# socket receive buffer size (should be at least the size of the
# in-kernel buffer (ipt_ULOG.o 'nlbufsiz' parameter)
rmem=131071

# libipulog/ulogd receive buffer size, should be > rmem
bufsize=150000

#####
# PLUGIN OPTIONS
#####

# We have to configure and load all the plugins we want to use
# general rules:
# 1. load the plugins _first_ from the global section
# 2. options for each plugin in separate section below

#
# ulogd_BASE.so - interpreter plugin for basic IPv4 header fields
#           you will always need this
plugin="/usr/lib/ulogd/ulogd_BASE.so"
```

```
# output plugins.  
plugin="/usr/lib/ulogd/ulogd_LOGEMU.so"  
#plugin="/usr/lib/ulogd/ulogd_OPRINT.so"  
#plugin="/usr/lib/ulogd/ulogd_MYSQL.so"  
#plugin="/usr/lib/ulogd/ulogd_PGSQL.so"  
#plugin="/usr/lib/ulogd/ulogd_SQLITE3.so"  
#plugin="/usr/lib/ulogd/ulogd_PCAP.so"
```

```
[LOGEMU]  
file="/var/log/ulogd/ulogd.syslogemu"  
sync=1
```

```
[OPRINT]  
file="/var/log/ulogd/ulogd.pktlog"
```

```
[MYSQL]  
table="ulog"  
pass="g3h31m"  
user="ulog"  
db="ulogd"  
host="localhost"
```

```
[PGSQL]  
table="ulog"  
schema="public"  
pass="g3h31m"  
user="ulog"  
db="ulogd"  
host="localhost"
```

```
[SQLITE3]  
table="ulog"  
db="/path/to/sqlite/db"  
buffer=200
```

```
[PCAP]  
file="/var/log/ulogd/ulogd.pcap"  
sync=1
```

Der Ulogd-Daemon unterstützt aktuell ein Input-Plugin, das IPv4-Pakete lesen und analysieren kann, und mehrere verschiedene Output-Plugins, mit denen Sie die Pakete in verschiedenen Formaten protokollieren können. Sie können zwischen den folgenden Output-Plugins wählen:

- MySQL
- PostgreSQL
- SQLite3
- Pcap-Datei
- LOGEMU (ähnlich Syslog-Protokollierung)
- OPRINT (einfache Protokollierung in einer Datei für Debugging)
- SYSLOG (protokolliert via Syslogd)

Die folgenden Input-Plugins können Sie nutzen:

- BASE (Grundfunktion, sollten Sie immer laden)
- LOCAL (protokolliert nur den Rechnernamen und den Zeitpunkt der Protokollierung)
- PWSNIFF (experimentelles Plugin für die Protokollierung der POP-3- und FTP-Kennwörter)

Sie müssen mindestens ein Input- und ein Output-Plugin wählen. Außerdem müssen Sie in der Konfigurationsdatei die folgenden Parameter setzen:

- `nlogroup=<gruppe>`: Hiermit wählen Sie die Gruppe, an die sich dieser Ulogd-Prozess binden soll. Jeder Ulogd-Prozess kann sich nur an eine Gruppe binden. Diese Zahl muss mit der Angabe in der Option `--ulog-nlogroup` des ULOG-Targets übereinstimmen. Wenn Sie verschiedene Gruppen in Ihren Regeln verwenden, benötigen Sie für jede Gruppe einen eigenen Ulogd-Prozess.
- `logfile=<datei>`: Dies ist die Protokolldatei des Ulogd-Daemons. Hier protokolliert der Daemon Fehler und Warnungen. Durch Angabe von `syslog` oder `stdout` protokolliert der Ulogd diese Informationen über den Syslogd oder auf der Standardausgabe.
- `loglevel=<level>`: Hiermit stellen Sie den Detailgrad der Ulogd-eigenen Protokollierung in dem `logfile` ein. Die Level sind in der Beispielkonfigurationsdatei angegeben.
- `plugin=<pfad/plugin>`: Mit diesem Parameter laden Sie die Plugins, die Sie verwenden möchten. Sie können diesen Parameter mehrfach verwenden. Die einzelnen Plugins benötigen teilweise zusätzliche Konfigurationsparameter (z.B. MySQL).
- `rmem=<größe>`: Mit diesem Parameter stellen Sie die Größe des Ulogd-Netlink-Socket-Empfangspuffers ein. Dieser muss mindestens der Größe des Netlink-Puffers des `ipt_ULOG`-Moduls entsprechen. Wenn Sie mit der Option `nobufsize=<größe>` diesen gesetzt haben, müssen Sie auch diesen Wert anpassen.
- `bufsize=<größe>`: Dies ist der eigentliche Ulogd-Empfangspuffer. Dieser muss mindestens die Größe des `rmmem`-Puffers haben.

Einzelne Plugins verfügen über zusätzliche Konfigurationsparameter. Bevor diese in eigenen Abschnitten besprochen werden, möchte ich kurz die Optionen erläutern, die Sie beim Start des Ulogd verwenden können:

- `-d, --daemon`: Daemon-Modus. Hiermit wechselt der Prozess in den Hintergrund.
- `-V, --version`: Anzeige der Version.
- `-h, --help`: Anzeige der Hilfe.
- `-c, --config <datei>`: Hiermit wählen Sie eine alternative Konfigurationsdatei aus. Die Default-Datei ist `/etc/ulog.conf`. Diese Option ist wichtig, wenn Sie mehrere Instanzen des Ulogd auf Ihrem System betreiben möchten. Das ist immer dann der Fall, wenn Sie in Ihren Regeln das `ULOG`-Target mit verschiedenen Netlink-Gruppen verwenden, um die Ausgabe in unterschiedliche Protokolle zu schreiben.

### 24.3.1 [OPRINT]

Dies ist ein sehr einfaches Ausgabemodul, das nur für Debugzwecke eingesetzt werden kann. Das Modul benötigt als einzigen Parameter den Dateinamen für die Ausgabe:

```
[OPRINT]
dumpfile=<datei>
```

### 24.3.2 [LOGEMU]

Dieses Modul protokolliert in eine Datei und emuliert dabei das Format der Syslog-Protokollierung ähnlich dem `LOG`-Target. Die Protokollierung erfolgt aber ohne Umweg über den Syslog direkt in die Datei. Sie müssen den Dateinamen angeben und können mit dem Parameter `sync` definieren, ob das Protokoll synchron (1) geschrieben werden soll (Default 0, asynchron). Dann tauchen neue Meldungen in dem Protokoll zu Lasten der Geschwindigkeit sofort auf.

```
[LOGEMU]
file=<datei>
sync=0|1
```

### 24.3.3 [MYSQL]

Hiermit protokollieren Sie in eine MySQL-Datenbank. Damit Sie dieses Plugin nutzen können, muss der Ulogd mit MySQL-Unterstützung übersetzt werden (`./configure --with-mysql`). Welche Informationen von diesem Plugin protokolliert werden, hängt von Ihrer MySQL-Tabelle ab. Dieses Plugin liest die verfügbaren Spalten der MySQL-Tabelle und vergleicht deren Namen mit den zur Verfügung stehenden Informationen. Als Beispiel für eine derartige Tabelle und die zu verwendenden Spalten-Typen können Sie auf die Datei `doc/mysql.table` zurückgreifen.

Wenn Sie an bestimmten Informationen nicht interessiert sind, können Sie durch Entfernen der Spalte dafür sorgen, dass dieses Plugin die Daten auch nicht protokolliert. Dadurch wird Ihre Datenbank schlanker. Wenn Sie die IP-Adresse als Zeichenkette in der Datenbank ablegen möchten, können Sie auch die Tabelle aus der

Datei `doc/mysql.table.ipaddr-as-string` verwenden. Dann müssen Sie aber auch Ulogd mit der Option `--with-mysql-log-ip-as-string` beim `./configure`-Aufruf übersetzen.

Das Modul benötigt die folgenden zusätzlichen Angaben:

- `table=<tabelle>`: Der Name der SQL-Tabelle.
- `ldb=<datenbank>`: Der Name der SQL-Datenbank.
- `host=<mysql_rechner>`: Der Rechner, auf dem der MySQL-Server läuft. Damit können Sie die Datenbank auf einem anderen Rechner unterbringen. Denken Sie nur daran, dass Ihre Firewall den Zugriff auf diesen Rechner erlauben muss.
- `port=<mysql_port>`: Der Port der MySQL-Datenbank.
- `user=<mysql_user>`: Der Benutzer, der für die Anmeldung an der Datenbank verwendet wird. Verwenden Sie aus Sicherheitsgründen bitte nicht das root-Konto.
- `pass=<mysql_password>`: Das Kennwort für die Anmeldung an der Datenbank.

#### 24.3.4 [PGSQL]

Dieses Output-Plugin arbeitet ähnlich wie das MySQL-Plugin. Damit das Plugin zur Verfügung steht, müssen Sie Ulogd mit der Option `--with-pgsql` bei der Übersetzung konfiguriert haben. Es verfügt über die gleichen Parameter, und Sie finden in der Ulogd-Distribution auch eine Beispiel-Tabelle `doc/pgsql.table`. Je nach PostgreSQL-Version müssen Sie jedoch einen zusätzlichen Parameter angeben: `schema=<schema>`. PostgreSQL ab Version 7.3 unterstützt diese Schemas. Damit können Sie datenbankübergreifende Views erzeugen. (Für Hintergrundinformationen über Schemas lesen Sie bitte auf <http://sql-info.de/postgresql/schemas.html> nach.) Wenn Sie PostgreSQL 7.3 oder eine neuere Version einsetzen, müssen Sie zusätzlich ein Schema angeben. PostgreSQL stellt für diese Zwecke das Default-Schema `public` zur Verfügung.

```
[PGSQL]
table="ulog"
schema="public"
pass="g3h31m"
user="ulog"
db="ulogd"
host="localhost"
```

#### 24.3.5 [PCAP]

Dieses Output-Plugin protokolliert das Paket im Libpcap-Format. Damit können Sie später die Protokolldatei in Ethereal oder Tcpcap öffnen und analysieren. Dies kann eine spätere forensische Analyse des Angriffs stark erleichtern. Dazu sollte dann aber von dem ULOG-Target auch das gesamte Paket protokolliert werden.

Das Plugin benötigt mindestens die Angabe der Pcap-Datei. Zusätzlich können Sie mit dem Parameter `sync` angeben, ob die Datei synchron (1) oder asynchron (0, Default) geschrieben werden soll.

```
[PCAP]
file="/var/log/ulogd/ulogd.pcap"
sync=1
```

### 24.3.6 [SQLITE3]

SQLite (<http://www.sqlite.org/>) ist eine kleine C-Bibliothek, die eine komplette, in andere Programme einbettbare SQL-Datenbank ohne zusätzliche Konfiguration darstellt. Sie benötigen keinen Dienst und müssen keinen Client oder Server konfigurieren. In vielen Umgebungen ist SQLite schneller als ein komplettes Datenbank-Managementsystem (DBMS). Um SQLite zu verwenden, müssen Sie Ulogd mit der Option `--with-sqlite` bei der Übersetzung konfiguriert haben.

Für die Erzeugung der Datenbank finden Sie in dem `doc/`-Verzeichnis der Distribution auch eine Datei `sqlite3.table`. Die einzigen Parameter, die Sie angeben müssen, sind der Pfad der Datenbank, der Name der Tabelle und die Größe des SQLite Puffers.

```
[SQLITE3]
table="ulog"
db="/path/to/sqlite/db"
buffer=200
```

### 24.3.7 [SYSLOG]

Das Syslog-Plugin protokolliert über den Syslog-Dienst. Damit sind die Protokollmeldungen identisch zu den Meldungen, die normalerweise das `LOG`-Target erzeugt. Sie müssen hier lediglich die Priorität der Meldung (Level, Priority) und die protokollierende Quelle (Facility) angeben.

```
[SYSLOG]
facility=LOG_KERN
level=LOG_INFO
```

## 24.4 Der Specter-Daemon

Der Specter-Daemon (<http://joker.linuxstuff.pl/specter/>) von Michal Kwiatkowski basiert auf dem Ulogd, wurde aber um einige Funktionen erweitert. Er wird genauso wie Ulogd unter der GPL-Lizenz vertrieben.

Zu den wesentlichen Unterschieden zum Ulogd zählen:

- Anderes Format der Konfigurationsdatei. Das Skript `ulogd2specter.pl` wandelt die Konfigurationsdatei `ulog.conf` in die Datei `specter.conf` um.
- Die Gruppierung mehrerer Netlink-Gruppen ist möglich. Sie müssen nicht für jede Netlink-Gruppe einen eigenen Prozess starten.
- Eine Gruppierung der Meldungen ist auch in Abhängigkeit von der Netfilter-Markierung möglich.

- Ein weiteres Output-Plugin `EXEC` kann externe Befehle aufrufen.
- Die Output-Plugins `MYSQL` und `PGSQL` unterstützen SSL für die Datenbankverbindung.
- Die Output-Plugins `SYSLOG` und `LOGEMU` unterstützen die Protokollierung der IP- und TCP-Optionen, der TCP-Sequenznummern und des MAC-Headers.

Specter befindet sich wie Ulogd in aktiver Entwicklung. Die aktuelle Version 1.4 wurde Juli 2005 veröffentlicht. Da es ansonsten in der Anwendung dem Ulogd gleicht und die Dokumentation sehr ausführlich ist, zeige ich hier nur eine Beispielkonfiguration mit einer kurzen Beschreibung:

```

plugins {
    BASE    /lib/specter/specter_BASE.so
    MYSQL   /lib/specter/specter_MYSQL.so
}

13 {
    :BASE
    :MYSQL
    db mydb
    host localhost
    user username
    pass password
    table ext_attacks
}

15 {
    :BASE
    :MYSQL
    db mydb
    host localhost
    user username
    pass password
    table int_attacks
}

```

In dem Block `plugins` definieren Sie die zu ladenden Plugins. Diese werden dann weiter unten konfiguriert. Die weiteren Blöcke beginnen mit der Netlink-Gruppe. Anschließend definieren Sie die zu verwendenden Plugins und geben zusätzliche Parameter an.

Wenn Sie nicht die Netlink-Gruppe, sondern die Firewall-Markierung für die Aufteilung der Meldungen verwenden möchten, müssen Sie zuvor noch einen Block `global` definieren:

```

global {
    grouping nfmark
}

```

```
n1group 1  
}
```

Nun nimmt Specter die Meldungen der Netlink-Gruppe 1 an und kann diese entsprechend der Firewall-Markierung aufteilen. Die Nummern vor jedem Block entsprechen nun nicht mehr der Netlink-Gruppe, sondern dieser Firewall-Markierung.