Tutorial/2: Logging und Timekeeping

Zeile für Zeile

Lukas Grunwald

An Log-Dateien erinnert man sich häufig erst, wenn etwas vorgefallen ist. Doch zu einem sicheren Netz gehört es nicht nur, den Betrieb regelmäßig zu analysieren, sondern auch, auf Vorfälle reagieren zu können.

Zum Absichern eines Systems genügt es nicht, dieses einmal auf einem hohen Sicherheitsniveau aufzusetzen und zu pflegen. Auch die regelmäßige Kontrolle, Beobachtung und Betriebsanalyse spielen eine wichtige Rolle, will man sich vor Angriffen schützen. Hierzu dienen die Systemmitteilungen und Log-Dateien, die Auskunft geben über den aktuellen Gesundheitszustand des Systems, also die Verfügbarkeit im Normalbetrieb oder das Fehlverhalten einer technischen Komponente, aber auch über den Sicherheitszustand des Systems im gefährdeten Bereich.

Solche Log-Einträge in den Dateien stammen vom Systemkern, von Diensten und Benutzerskripten. Sie dienen der Erkennung und Auswertung von Vorfällen wie Angriffen auf das System, Portscans und Exploits ebenso wie von Ereignissen des allgemeinen Betriebs, etwa Verbindungsaufnahmen. Dadurch kann der Sicherheitsadministrator aktuelle Bedrohungsszenarien erkennen und daraufhin das System schützen, es gegebenenfalls vom Netz nehmen oder aktualisieren.

Unter Unix/Linux steht schon seit Jahren eine Methode der Ereignisprotokollierung zur Verfügung, das so genannte Syslog oder System-Logging. Das Syslog-System unterscheidet unterschiedliche Log-Level (Dringlichkeiten) und Log-Facilities (Zugehörigkeiten (siehe „Log-Level“ und -Facilities). Je nach Level und Facility geben klassische Systeme die Benachrichtigungen auf der Konsole aus, tragen sie in eine Log-Datei ein oder senden sie per E-Mail an die Administratoren. Bei einer großen Installation mit vielen Servern – etwa in einem Rechenzentrum – ist das aber nicht mehr handhabbar.

Um die Daten für eine Auswertung an zentraler Stelle zu sammeln, empfiehlt es sich, einen Log-Server zu betreiben. Seine Aufgabe ist es, die Daten vorzufiltern, das heißt, die sicherheitsrelevanten Daten soweit zusammenzuschrumpfen, das sie nur noch nach sicherheitskritischen respektive betriebskritischen Kriterien auszuwerten sind. Das Zusammenstellen solcher Auswertungskriterien kann recht aufwendig sein. Sie lassen sich nur im Regelbetrieb erstellen, indem man häufig auftretende Log-Ereignisse – den so genannten Common-Case – herausfiltert und Filterregeln für Meldungen über ungewöhnliche Ereignisse erarbeitet.

Ein weiterer Anwendungszweck eines Syslog-Servers ist es, die Log-Dateien einem Auditor zur Verfügung zu stellen – also einer Person aus der Revision, die regelmäßig den ordnungsgemäßen Betrieb der Server zu überwachen hat. Zu diesem Zweck transportiert der Log-Server die Dateien, die so genannten Audit-Trails, an einen weiteren Security-Auditation-Rechner, beispielsweise über einen SSH-Tunnel. Dort werden dann Audit-Listen erstellt, die etwa die Root-Logins der Administratoren protokollieren und anhand derer man das Verhalten der Administratoren auf den Systemen nachvollziehen kann.

Zentrale Sammelstelle

Zwar unterstützt das klassische Syslog-Format das Weiterleiten an einen Log-Server. Allerdings gehen die Syslog-Meldungen als ungesicherte UDP-Datagramme übers Netz, was nicht besonders zuverlässig ist, da UDP nicht gegen den Verlust der Datagramme schützt. Zudem ist UDP nicht gegen Spoofing gesichert, also gegen das Fälschen respektive Ersetzen durch nichtssagende oder den Log-Server irritierende Meldungen. Man benötigt also einen neuen Mechanismus, das so genannte Secure Syslogging.

Auf dem Beispielserver soll das Syslog-NG – Syslog Next Generation – die Arbeit verrichten (www.balabit.com/products/syslog_ng). Zwar existieren noch andere Secure-Syslog-Programme, die eine solche Aufgabe ebenfalls erfüllen, aber gegenüber Syslog-NG einige Nachteile aufweisen. Ein Beispiel ist das SDSC (Secure Syslog), das ebenfalls auf Protokoll-Level kompatibel mit Syslog „Classic“ (514/UDP) ist (security.sdsc.edu). Allerdings ist es noch nicht lange in der Praxis erprobt und durch sein übertriebenes Layering sowie seinen wissen-

schaftlichen Ansatz wesentlich schwerer zu handhaben als Syslog-NG.

Syslog-NG unterstützt neben UDP auch TCP und bietet trotz einfacher Handhabung viele unterschiedliche Log-Mechanismen, etwa das Weitergeben von Ereignissen an eine SQL-Datenbank. Ebenso kann es die Meldungen an Programme via Pipe oder durch Aufruf etwa eines SNMPv3-Inform-Sender weiterleiten. Eine weitere Stärke von Syslog-NG liegt in der Flexibilität der eingebauten Filter.

Als Quelle für einen Log-Server dient nicht nur der Beispiel-Webserver des ersten Tutorial-Teils. Der zentrale Log-Server kann auch andere Log-Ereignisse erfassen, etwa die der Firewall, des Host- und des Netz-basierten Intrusion Detection Systems, ebenso die der so genannten Sensoren, der Datenbanken, der anderen Serversubsysteme und Teilkomponenten der DMZ (Demilitarisierte Zone). Dadurch kann der Admin die Wechselbeziehungen aller Ergebnisse herstellen und einen permanenten Überblick über den Systemzustand und den Sicherheitszustand der gesamten Internet-/Intranet-Zone behalten.

Alles eine Frage der Zeit

Damit man eine Korrelation erfolgreich durchführen kann, müssen die Systemuhren aller beteiligten Rechner mit einander und einer zuverlässigen externen Zeitquelle sychronisiert und möglichst auf die UTC eingestellt sein (siehe „NTP-Dämonen und Timekeeping“).

Eine falsche Uhrzeit auf einem sicherheitskritischem Server wäre fatal, wenn beispielsweise ein Angriff über einen Einwahl-Zugang eines ISPs stattfindet und der Abgleich der benutzten IP-Adresse zu einem Unschuldigen führt, der plötzlich Besuch von der Polizei bekommt. Auch für die revisionssichere und vor einem Gericht verwertbare Dokumentation sowie den ordnungsgemäßen Betrieb ist die Synchronisation zwingend notwendig.

Nachdem nun die Systemzeit der Server auf einen einheitlichen Stand gebracht ist, folgt die Konfiguration des Syslog-NG-Systems, und zwar durch Editieren der Datei */etc/syslog-ng.conf* respective */etc/syslog-ng/syslog-ng.conf* (siehe Listing 2 und 3). Zuerst sind die Quellen mit *source* und Ziele mit *destination* zu definieren. Die Syntax verlangt immer das Muster *source <src-name> { src-driver(parameter); src-driver(parameter); };* respektive *destination <dest-name> { dest-driver(parameter); dest-driver(parameter); };*.

Source-Driver können sein: *file*, *pipe*, *interna*, *unix-dgram* (BSD), *unix-stream* (Linux), *udp* oder *tcp*, Destination-Driver zusätzlich *usertty* und *program*, aber kein *internal*. Hinzu kommen die optionalen Filterregeln, immer in der Syntax *filter <filtername> { funktion(expression); funktion(expression); };*.

Syslog-NG kennt wie der klassische Syslog-Dämon fest definierte Log-Level (Dringlichkeiten) und -Ressourcen (facilities), nach denen er die Meldungen sortieren und zustellen kann (siehe „Log-Level und -Facilities“). Da die Log-Level feste Prioritäten haben, lassen sich Dringlichkeitsbereiche, etwa *level(info (siehe „“ warn)*, ebenso wie Listen wie *facility(auth, authpriv, cron)* definieren. Zusätzlich zu *level* und *facility* kennt Syslog-NG die Filterfunktionen *program(regexp)*, *host (regexp)* und *match(regexp)*. Sie alle lassen sich mit *and*, *or* und *not* verknüpfen.

Der Definition der geeigneten Quellen, Ziele und Filter folgen die Log-Statements oder Log-Regeln: *log { source(src-name); filter(filtername); filter(filtername); destination(dest-name); };*, wobei für die Filterverknüpfung immer ein logisches „und“ gilt.

Überblick gefragt

Da auf jedem einzelnen Rechner ein Syslog-NG-Dämon laufen muss, sind auf jedem einzelnen die *syslog-ng.conf*-Dateien an das jeweilige System anzupassen. Die mit den Syslog-NG-Paketen installierte beispielhafte Konfigurationsdatei ist recht umfangreich gehalten und gehört als erstes abgespeckt. Sämtliche Dienste, die nicht auf dem System laufen, sind aus den Definitionen zu entfernen. Andere dagegen, die die Hauptarbeit auf dem jeweiligen Sys-

X-TRACT

- Um über die Vorgänge im eigenen Netz auf dem Laufenden zu bleiben, genügt es nicht, ab und zu einen Blick in die Standard-Log-Dateien zu werfen.
- Für eine effektive Auswertung der Ereignisse ist ihre Korrelation und die Abstimmung der Systeme, etwa ihrer Systemzeit, von entscheidender Bedeutung.
- Neuere Secure-Syslog-Systeme stellen Filter und Schnittstellen wie Programmaufrufe zur Verfügung, um die Weiterleitung und Auswertung der Informationen zu vereinfachen.
- Logging-Systeme lassen sich auch für das Ingangsetzen von Alarmierungsketten einsetzen mit anderen Überwachungssystemen verknüpfen.

Tutorial-Übersicht

Teil 1: Härten und Abspecken eines sicherheitsrelevanten Linux-Systems am Beispiel eines minimalen statischen Webservers.

Teil 2: Erstellen von Ereignismitschnitten und Auswertung von Log-Dateien.

Teil 3: Konfigurieren und Härten einer Firewall am Beispiel von Linux. Planung, Dokumentation und Aufbau von Audit-Kreisläufen.

Log-Level und -Facilities

Log-Level (Dringlichkeiten):

emerg (panic)	System ist unbrauchbar
alert	Erfordert einen Eingriff
crit	Kritischer Zustand
err (error)	Fehlerfall
warning (warn)	Warnung
notice	Wichtiges Ereignis
info	Zur Information
debug	Umfangreiche Meldungen

Log-Facilities:

auth (security)	Security/Authorisations-Nachrichten
authpriv	Private Security/Authorisations-Nachrichten
cron	Clock-Dämon (cron, at)
daemon	System-Dämonen ohne eigene Facility
ftp	FTP-Dämon
kern	Kernel-Messages
local0 – local7	Reserviert für den lokalen Gebrauch
lpr	Lineprinter-Subsystem
mail	Mail-Subsystem
news	News-Subsystem
syslog	Vom syslogd intern generierte Nachrichten
user	Generische User-Level-Messages (default)
uucp	UUCP-Subsystem

NTP-Dämonen und Timekeeping

Nicht nur für das Syslogging, auch für andere Applikationen wie Mess-Anwendungen oder große Datenbanksysteme benötigt man eine exakte und synchronisierte Systemzeit auf seinen Rechnern. Dafür sorgt das Network Time Protocol (NTP), das die Uhren untereinander oder noch besser mit einer genauen Zeitreferenz synchronisiert (www.ntp.org). Leider gibt es historisch bedingt häufig Konfusionen über Namen und Versionen des NTP sowie seiner Implementierungen. Als Standard hat sich NTPv3 (RFC 1305) durchgesetzt. Zwar ist – zumindest unter FreeBSD – das alte Time-Protokoll noch im Einsatz, das eignet sich aber nicht für die Anbindung an externe Zeitquellen und arbeitet sehr ungenau.

Der Nachfolger NTPv4 – zu finden auf www.ntp.org – gilt als künftige Referenzimplementierung, ist aber nicht in einer RFC formalisiert. Er bringt neben einer Anpassung an die IPv6- und OSI-Adressierung einige Verbesserungen mit: Sie reichen vom Redesign des „clock discipline algorithm" über die Unterstützung neuer Plattformen, Betriebssysteme und „Nanokernel" bis hin zu neuen Treibern für Referenzuhren und einer verbesserten kryptografischen Public-Key-Unterstützung.

Daneben existiert die vereinfachte Variante SNTP (Simple Network Time Protocol), das in der Version 4 (RFC 2030) sowohl IPv4 als auch IPv6 unterstützt. SNTP verwendet zwar dasselbe Datagramm- und Zeitstempelformat wie NTP, benutzt aber intern einen anderen Abgleichsalgorithmus, der einfacher zu implementieren ist und eine geringere Genauigkeit erreicht.

Listing 1: /etc/ntp.conf

```
# NTP configuration file (ntp.conf)
#
# Ignoriere alle Requests
restrict default ignore
#
# Erlaube Administration via ntpq
restrict 193.108.181.6
restrict 127.0.0.1
#
# Clients
restrict 192.168.181.0 mask 255.255.255.0 nomodify
restrict 213.252.143.127 mask 255.255.255.128 nomodify
#
#
restrict 192.53.103.103          # ntp1.ptb.de
restrict 192.53.103.104          # ntp2.ptb.de
#
# Server und Peers
server 192.53.103.103          # ntp1.ptb.de
server 192.53.103.104          # ntp2.ptb.de
peer 10.0.0.1
peer 10.0.0.2
peer 10.0.0.3
#
# local clock is unsynchronized restrict
fudge 127.127.1.0 stratum 16
#
# Miscellaneous stuff
#
driftfile /var/state/ntp.drift # path for drift file
statsdir /var/log/ntp  # directory for statistics files
filegen peerstats file peerstats type day enable
filegen loopstats file loopstats type day enable
filegen clockstats file clockstats type day enable
#
# Authentication stuff
#
keys /etc/ntp/ntp.keys     # path for keys file
trustedkey 3 4 5 6      # define trusted keys
requestkey 1 # key (1) for accessing server variables
controlkey 2 # key (2) for accessing server variables
```

Als die Entwickler vor 10 Jahren mit dem Redesign des NTP und dem Einfügen neuer Funktionen begannen, kennzeichneten sie diese Erweiterungen durch ein „X" als Experimental. Offiziell gibt es schon lange kein XNTP (Experimental Network Time Protokoll) mehr, sondern nur noch schlicht das NTP. Wer heute ein *xntp*-Paket auf einer Linux- oder Unix-Distribution antrifft, hat es entweder mit einer veralteten oder mit einer falsch benannten Version zu tun. Es sollten nur noch NTP und SNTP in Gebrauch sein.

Die Zeit aus der Maschine

Als internationaler Standard für die Zeitmessung gilt eine koordinierte Zeitskala namens UTC (Universal Time Coordinate, Koordinierte Weltzeit), deren Einheit Sekunde (SI) auf einer atomaren Schwingung des Cäsium-133-Isotops basiert. Diese Atomzeitskala löste 1972 die Greenwich Mean Time (GMT) ab. Sie führte gleichzeitig das Einfügen von Schaltsekunden ein, sobald sie mehr als 0,9 Sekunden von der astronomischen Zeit UT1 abweicht.

UTC stellt zudem die Referenz für alle Quellen von NTP-Servern dar, etwa den Atomuhren der Physikalisch-Technischen Bundesanstalt (PTB) in Braunschweig (www.ptb.de). Letztere stellt das Zeitsignal via Internet (ntp1.ptb.de, ntp2.ptb.de), Modemzugang (0531-512038) oder Radiosignal (DCF-77) für den Einzugsbereich zur Verfügung (www.dcf77.org). Auch die Télé Distribution Française (TDF) unterhält einen Radiosender mit Zeitsignal. Die Atomuhren der GPS-Sateliten benutzen dagegen die astronomische Zeitskala UT1; ihre Empfänger konvertieren aber die Zeitmessung in UTC.

NTP kennt Güteklassen der Zeitquellen, die so genannten „Stratum Level", die die Hops zur Referenzuhr darstellen: Stratum 0 ist die Referenzuhr selbst, die immer an eine Atomuhr gekoppelt ist, Stratum 1 bezeichnet die Zeitserver, die eine direkte Anbindung an eine Referenzuhr besitzen, Stratum-2-Server holen ihre Zeit wiederum von einem oder mehreren Stratum-1-Servern.

Der NTP-Dämon lässt sich aber nicht nur als Client oder Server, sondern auch als Peer mit anderen Servern auf gleicher oder unterschiedlicher Stratum-Ebene betreiben. Damit erhöht man die Ausfallsicherheit und schafft eine bessere Berechnungsgrundlage: Die Peers gleichen sich untereinander ab und errechnen auf Basis des Stratum-Werts, der Abweichung und der Drift (s. u.) eine gemittelte Zeit, auch wenn die Referenzuhr einmal ausfallen sollte. Durch die Statistik der Drift kann der NTP-Dämon außerdem die Vor- oder Nacheilzeit eines Systems berechnen. Als Server stellt der NTP-Dämon seine eigene Zeit im Netzwerk zur Verfügung – entweder auf Anfrage oder per Unicast, Manycast oder Multicast/Broadcast.

Für die Clientseite bieten die NTP-Pakete zwei Tools an: *ntpdate* und *ntpd*. *ntpdate* fragt einen NTP-Server einmalig ab und stellt nach dem empfangenen Zeitstempel die Systemuhr ein. Es ist allerdings keine gute Idee, einfach *ntpdate* stündlich als Cron-Job laufen lassen. Da *ntpdate* bei einem Offset größer als ±128 ms in einer Hau-Ruck-Methode per *settimeofday()* einfach die Systemzeit umstellt, ergeben sich Sprünge und es lässt sich nicht nachvollziehen, um wie viel schneller oder langsamer die Systemuhr läuft (Drift). Auch achtet *ntpdate* nicht auf falsche Zeitstempel. *ntpd* dagegen veranlasst den Kernel, die Softwaresystemuhr geringfügig langsamer beziehungsweise schneller laufen zu lassen, um die Zeit anzugleichen, damit es beim Korrigieren der Systemzeit nicht zu Sprüngen kommt. Wenn er von einer Radio-Clock einen vollkommen abweichenden Zeitstempel empfängt, stellt er diese Uhrzeit nicht ein.

Timebandit mit Adressbuch

Seine Konfiguration holt sich der *ntpd* aus der Datei */etc/ntp.conf* (siehe. Listing 1). Die Syntax folgt immer dem Muster <*keyword*> <*address/location*> <*option*>. Die synchronisierten Quellen wie Zeitserver und eingebaute Empfänger sind mit *server* anzugeben, Peers mit *peer*, erlaubte Zugriffe mit *restrict*. Die Zeile *restrict default ignore* sorgt dafür, dass der *ntpd* Zugriffe aus nicht aufgeführten Netzen, Netzsegmenten oder Hosts abblockt.

Eine Besonderheit stellen die Pseudo-IP-Adressen dar: *127.127.1.0* adressiert die lokale Systemuhr (local clock). Sie sollte allerdings nur als nicht synchronisierte und mit hohem Stratum-Wert aufgeführt sein. Dafür sorgt der Eintrag *fudge 127.127.1.0 stratum 16*. *127.127.x.n* adressiert lokale Zeitgeber wie Funkuhren. Zuvor muss aber ein symbolischer Link */dev/refclock-n* erzeugt werden, der auf das Gerät zeigt. Das *x* zeigt den Gerätetyp an. Üblicherweise verwendet man die Adresse *127.127.8.0*, die über den Link */dev/refclock-0* auf die DCF77-Uhr am ersten seriellen Port */dev/ttyS0* verweist.

Der Eintrag *driftfile* gibt Pfad und Name der Datei für die Drift-Statistik an, *statsdir* den Pfad für die vom *ntpd* generierten Statistikdateien. Das Schlüsselwort *filegen* weist den Dämon an, die fehlenden Dateien zu generieren.

Wer sich vor einer Verfälschung der Zeitstempel schützen will, sollte sie auf jeden Fall kryptographisch sichern, da NTP immer UDP als Transport-Protokoll verwendet. NTP bietet, wenn es gegen die SSL-Library gelinkt ist, sowohl symmetrische als auch asymmetrische Keys an: Der Befehl *ntp-genkeys –g r* erzeugt einen RSA-Key, *ntp-genkeys –g m* eine MD5-Hash-Datei. Die erzeugten Keys lassen sich nun auf allen Peers und Servern im lokalen NTP-Netz installieren. Zum Testen, ob der NTP-Server ordnungsgemäß funktioniert, kann der Befehl *ntpq -p* dienen. Er gibt eine Tabelle mit den erreichten Servern und Peers und den übermittelten NTP-Daten aus. Analog zu *traceroute* lassen sich mit *ntptrace* die Quellen für den NTP-Server zurückverfolgen.

tem verrichten, wie ein *httpd* auf einem Webserver oder ein *smtpd* auf einem Mailserver, und für die es gegebenenfalls einen zuständigen Admin gibt, verdienen besondere Beachtung. Beispielsweise könnte es ratsam sein, ihre Logs je nach Level in unterschiedliche Dateien zu schreiben oder alle ihre Einträge ab einer bestimmten Dringlichkeitsstufe auf einer Konsole respektive der Root-Konsole auszugeben.

Regelwerk

Für den Mail-Dämon etwa genügt es, zwei Filterregeln wie *filter f_mail { facility(mail); };* und *filter f_err { level(warn .. emerg); };* durch das Log-Statement *source(src); filter(f_mail); filter(f_err); destination(console_all)* miteinander zu kombinieren. Für Dämonen, für die keine eigene Facility definiert ist, eignet sich die Filterfunktion *match*. Um etwa alle von *httpd* gemeldeten Ereignisse herauszufiltern, empfiehlt sich die Regel *filter f_httpd { facility(daemon) and match(„httpd: „); };*. Danach lassen sich die unterschiedlichen *httpd*-Meldungen durch Level-Filter weiter auffächern.

Will man mit einem eigenen Benutzerskript Meldungen direkt über das Syslog-System absetzen, eignet sich der Befehl *logger*. Er generiert über Aufrufe von Syslog-Funktionen Einträge ins System-Log, die der *syslogd* dann je nach Priorität an die entsprechende Ausgabestelle leitet.

Eine Besonderheit des Syslog-NG besteht darin, dass er den Linux-Kernel-Logdämon *klogd* ebenfalls ersetzen kann, indem er aus */proc/kmsg* liest. Wenn man allerdings *pipe(/proc/kmsg)* in der Source-Definition aufführt, sollte man auf jeden Fall vorher sicherstellen, dass der *klogd* deaktiviert ist. Außerdem empfiehlt es sich, auf allen Systemen, die ihre Logs an den Syslog-Server ausliefern, nur lokale Logquellen zuzulassen wie etwa *unix-stream(„/dev/log“);* und *internal();*, die interne Syslog-NG-Logquelle.

Nur beim Syslog-Server selbst darf die Angabe der Netzwerk-Schnittstelle nicht fehlen. Befinden sich noch Hosts mit dem klassischen Syslog-System im Netz, sollte – um die Kompatibilität zu gewährleisten – UDP erlaubt sein, für alle anderen TCP. Die Angabe der Optionen *ip* und *port* bindet den Syslog-NG-Dämon an ein bestimmtes Interface und einen Port.

Listing 2: /etc/syslog-ng.conf auf dem Syslog-Client

```
# First, set some global options.
#
options {
use_fqdn(yes);
       use_dns(no);
       keep_hostname(yes);
       long_hostnames(off);
       sync(1);
       log_fifo_size(1024);
};

# Logs may come from unix stream,
# but not from another machines.
source src {
       unix-stream("/dev/log");
       internal();
       pipe("/proc/kmsg");
};

# destinations
#
# Standard logfile und `catch-all' logfiles
destination auth { file("/var/log/auth.log"); };
destination syslog { file("/var/log/syslog"); };
destination kern { file("/var/log/kern.log"); };
destination cron { file("/var/log/cron.log"); };
destination mess { file("/var/log/messages"); };
destination errors { file("/var/log/err.log"); };
destination all { file("/var/log/all.log"); };

# Logging for the mail subsystem
destination mail { file("/var/log/mail/mail.log"); };
destination mailinfo { file("/var/log/mail/mail.info"); };
destination mailwarn { file("/var/log/mail/mail.warn"); };
destination mailerr { file("/var/log/mail.err"); };
destination mailalert { file("/var/log/mail.alert"); };

# consoles, scripts and syslog server
destination c_root { usertty("root"); };
destination c_all { file("/dev/tty8"); };
destination loghost { tcp(ip(10.0.0.1) port(514)
       keep-alive(yes)); };
destination sms { program("/usr/bin/sendsms 01711234567 \
       LOG_SERVER_DOWN"); };

# filter options
#
# facility filters
filter f_authpriv { facility(auth, authpriv); };
filter f_syslog { facility(syslog); };
filter f_cron { facility(cron); };
filter f_daemon { facility(daemon); };
filter f_kern { facility(kern); };
filter f_mail { facility(mail); };
filter f_user { facility(user); };

# loglevel filters
filter f_alert { level(crit .. emerg); };
filter f_err { level(err); };
filter f_warn { level(warn); };
filter f_info { level(info, notice); };

# other filters
filter f_mess { level(info .. warn)
       and not facility(auth, authpriv, mail); };
filter f_cmail { level(notice .. emerg)
       and facility(mail); };
filter f_cother { level(err .. emerg)
       and not facility(auth, authpriv, kern, mail); };
filter f_nosyslog  { match(syslog-ng)
       and match(Connection broken); };

# log statements actually send logs somewhere
#
log { source(src); filter(f_authpriv); destination(auth); };
log { source(src); filter(f_syslog); destination(syslog); };
log { source(src); filter(f_cron); destination(cron); };
log { source(src); filter(f_daemon); destination(mess); };
log { source(src); filter(f_kern); destination(kern); };
log { source(src); filter(f_user); destination(mess); };
log { source(src); filter(f_mail); destination(mail); };
log { source(src); filter(f_mail); filter(f_info);
       destination(mailinfo); };
log { source(src); filter(f_mail); filter(f_warn);
       destination(mailwarn); };
log { source(src); filter(f_mail); filter(f_err);
       destination(mailerr); };
log { source(src); filter(f_mail); filter(f_alert);
       destination(mailalert); };
log { source(src); filter(f_mess); destination(mess); };
log { source(src); filter(f_err); destination(errors); };
log { source(src); destination(all); };
log { source(src); filter(f_alert); destination(c_root); };
log { source(src); filter(f_cother); destination(c_all); };
log { source(src); filter(f_cmail); destination(c_all); };
log { source(src); filter(f_kern); destination(c_all); };

# send everything to loghost, too
log { source(src); destination(loghost); };

# SMS to admin if syslog server down:
log { source(src); filter(f_nosyslog); destination(sms); };
```

Um sicherzustellen, dass die Rechner mit Syslog-NG nur TCP benutzen, genügt es, in der Zieldefinition nur TCP anzugeben, etwa *destination loghost { tcp(ip(10.0.0.10) port(514) keep-alive(yes)); };*.

Auf jeden Fall gilt es zu verhindern, dass Ereignisse nicht nachvollziehbar weitergeleitet werden oder jemand durch Spoofing dem Log-Server Syslog-Meldungen unterschiebt. Daher sollte man bei Systemen mit erhöhter Sicherheit ein gesondertes Log- und Timekeeping-Netz ohne Zugang zum Internet installieren. Dieses Netz kann als VLAN oder als separates Admin-LAN aufgebaut sein. Solche Out-Band-Management- und -Logging-Methoden erhöhen die Sicherheit des Netzes erheblich.

Inwieweit auf den einzelnen Hosts die Logs auch noch lokal vorgehalten werden und wohin der Syslog-Server die eingegangenen Logs schreibt, hängt von der firmeninternen Struktur und den zuvor erstellten Adminstrationsvorgaben ab. Statt alle eingehenden Mitteilungen in Log-Dateien zu sortieren, besteht die Möglichkeit, mit ihnen über ein Skript eine SQL-Datenbank zu füttern, über die dann eine Auswertung erfolgt.

Zur Archivierung können die Logs zusätzlich auf einem gesonderten Speichersystem abgelegt werden, beispielsweise nach Hosts sortiert mit *destination std { file(„/var/log/HOSTS/$HOST/$YEAR/$MONTH/$DAY/$FACILITY_$HOST_$YEAR_$MONTH_$DAY“*, wobei */var/log/HOSTS* der Mountpoint etwa für ein DVD-RAM-, ein MO-, ein WORM-Laufwerk oder ein RAID-System ist.

Rissfeste Alarmkette

Daneben muss die Alarmierungskette bei Notfällen und sicherheitsrelevanten Ereignissen – die klassifiziert und in einem Notfallhandbuch dokumentiert sein sollten – so angepasst sein, dass Syslog-NG als Anstoßmechanismus dient. Bei unberechtigten Zugriffsversuchen oder anderen Vorfällen kann ein Mail- oder SMS-Alert-Skript

Listing 3: /etc/syslog-ng.conf auf dem Syslog-Server

```
# First, set some global options.
options {
        use_fqdn(yes);
        use_dns(no);
        dns_cache(no);
        keep_hostname(yes);
        long_hostnames(off);
        sync(1);
        log_fifo_size(1024);
};

# Logs may come from unix stream or from another machines
#
source src {
        pipe("/proc/kmsg");
        unix-stream("/dev/log");
        internal();
};
source all {
        pipe("/proc/kmsg");
        unix-stream("/dev/log");
        internal();
        udp ()
        tcp(ip(10.9.9.3) port(514) keep-alive(yes));
};

# destinations.

# standard and `catch-all' logfiles.
destination auth { file("/var/log/auth.log"); };
destination syslog { file("/var/log/syslog"); };
destination cron { file("/var/log/cron.log"); };
destination daemon { file("/var/log/daemon.log"); };
destination kern { file("/var/log/kern.log"); };
destination lpr { file("/var/log/lpr.log"); };
destination mail { file("/var/log/mail.log"); };
destination uucp { file("/var/log/uucp.log"); };
destination debug { file("/var/log/debug"); };
destination mess { file("/var/log/messages"); };
destination all_local { file("/var/log/all.log"); };

# consoles and scripts
destination c_root { usertty("root"); };
destination c_local { file("/dev/tty8"); };
destination xconsole { pipe("/dev/xconsole"); };
destination mail-alert {
        program("/usr/local/bin/syslog-mail-perl"); };
destination swatch { program("/usr/bin/swatch \
        --read-pipe=\"cat /dev/fd/0\""); };
destination sqlsyslogd {
        program("/usr/local/sbin/sqlsyslogd -u sqlsyslogd \
        -t logs sqlsyslogd -p"); };

# automatic host sorting on loghost
destination std {
        file("/var/log/HOSTS/$HOST/$YEAR/$MONTH/$DAY/$FACI
LITY_$HOST_$YEAR_$MONTH_$DAY" owner(root) group(root) \
        perm(0600) dir_perm(0700) create_dirs(yes)); };

# filter options
#
#facility filters
filter f_authpriv { facility(auth, authpriv); };
filter f_syslog { facility(syslog); };
filter f_cron { facility(cron); };
filter f_daemon { facility(daemon); };
filter f_kern { facility(kern); };
filter f_lpr { facility(lpr); };
filter f_mail { facility(mail); };
filter f_user { facility(user); };
filter f_uucp { facility(uucp); };
filter f_public { not facility(auth, authpriv); };

#level filters
filter f_debug { not facility(auth, authpriv, kern)
        and level(debug); };
filter f_mess { level(info .. warn)
        and not facility(auth, authpriv); };
filter f_err { level(err, crit); };
filter f_emergency { level(alert, emerg); };

# other filters
filter f_attack_alert { match("attackalert"); };
filter f_ssh_login_attempt { program("sshd.*")
        and match("(Failed|Accepted)")
        and not match("Accepted (hostbased|publickey) for \
        (root|zoneaxfr) from (10.4.3.1)"); };

# log statements actually send logs somewhere
#
# logs from localhost
log { source(src); filter(f_authpriv); destination(auth); };
log { source(src); filter(f_syslog); destination(syslog); };
log { source(src); filter(f_cron); destination(cron); };
log { source(src); filter(f_daemon); destination(daemon); };
log { source(src); filter(f_kern); destination(kern); };
log { source(src); filter(f_lpr); destination(lpr); };
log { source(src); filter(f_mail); destination(mail); };
log { source(src); filter(f_uucp); destination(uucp); };
log { source(src); filter(f_debug); destination(debug); };
log { source(src); filter(f_user); destination(mess); };
log { source(src); filter(f_mess); destination(mess); };
log { source(src); destination(all_local); };
log { source(src); filter(f_emergency); filter(public);
        destination(c_local); };
log { source(src); filter(f_err); filter(public);
        destination(c_local); };

# all logs
#
log { source(all); filter(f_emergency);
        destination(c_root); };
log { source(all); filter(f_err); destination(c_root); };

# slap it into a MySQL database
log { source(all); destination(sqlsyslogd); };

# find messages with "attackalert" in them
log { source(all); filter(f_attack_alert);
        destination(mail-alert); };

# find messages reporting attempted ssh logins
log { source(all); filter(f_ssh_login_attempt);
        destination(mail-alert); };

# send all logs to swatch for (near) real-time alerts
log { source(all); destination(swatch); };

# automatic host sorting
log { source(all); destination(std); };
```

auf dem Syslog-Server den Admin informieren. Parallel dazu könnte der Syslog-NG alle Logs an einen Watchdog wie *swatch* (www.stanford.edu/~atkins/swatch) weiterleiten, der bei bestimmten Vorfällen die Alarmierungskette in Gang setzt (siehe Listing 3, vorletzter Eintrag).

Sind solche Anstoßmechanismen auf dem Syslog-Server installiert, sollte auf den einzelnen Hosts zumindest eine Notfall-Regel für den Fall eingebaut sein, dass der Logserver ausfällt oder nicht erreichbar ist. Bei der Übertragung der Logs mit TCP und Keep-Alive gibt der Syslog-NG-Dämon automatisch eine Benachrichtigung aus, wenn die Verbindung reißt. Ist eine entsprechende Regel definiert, kann er über ein SMS-Modul den Admin sofort darüber informieren, dass die Logging-Kette abgerissen ist (siehe Listing 2, Filter f_nosyslog).

Eine weitere Methode der Alarmierung besteht darin, parallel zum sicheren Weiterleiten von Log-Meldungen an den Syslog-Server einen so genannten SNMPv2-Trap- oder SNMPv3-Inform-Sender (Simple Network Management Protocol) auszuführen. Beide leiten die Nachricht über einen sicherheitsrelevanten Vorfall umgehend an ein Netzwerkmanagementsysem (NMS) wie HP Openview, IBM Tivoli oder Cisco Works weiter. Der Vorteil der Version 3 des SNMP gegenüber der Version 2 besteht darin, dass sie eine Verschlüsselung der Nachricht unterstützt.

Auch kann man den umgekehrten Weg gehen, indem man mit einem SNMP-Aggregator respektive -Translator die SNMP-Traps von Netzwerkgeräten wie Switches, Hubs oder Druckern direkt in das Syslog-Protokoll übersetzt und sie damit in das netzweite Logging-System einbindet. *snmptrapfmt* etwa übersetzt die Traps in ein für Log-Dateien geeignetes Format und leitet sie in eine Log-Datei oder an den Syslog-Dämon weiter.

Wenn der Verdacht auf einen momentan laufenden Angriff besteht, kann sich ein Mittschnitt des Netzwerkverkehrs lohnen. Mit dem Befehl *tcpdump –i eth0 –w /var/run/allmytraffic.dump* landen sämtliche Netzwerkpakete, die von dem Interface *eth0* aus zu lesen sind, in der angegebenen Datei.

Aber Vorsicht: wenn Benutzer etwa bei einer Telnet-Session Klartext-Passwörter übertragen, landen die ebenfalls im Dump. Bei einem unberechtigten Zugriff auf die Dump-Datei würde man damit einem Angreifer die Passwörter auf dem silbernen Tablett präsentieren. Befindet sich ein Switch mit einem Readonly-Mirror-Port im Netz, kann man vom Admin-Netz aus darüber den Netzwerkverkehr mitschneiden und den Dump für eine spätere Auswertung sicher verwahren. Zudem kann man den Dump nutzen, um nach unsicheren Klartext-Passwörtern und anderen gefährlichen Daten im Netz zu suchen.

Fazit

Es gibt keine Instant-Lösung, um den Netzwerkbetrieb zu kontrollieren und zu beobachten. Man kann nur mit vielen mächtigen Werkzeugen eine Organisation aufbauen, die möglichst zuverlässig den Verantwortlichen je nach Situation warnt beziehungsweise alarmiert. Ein solches Frühwarnsystem ist inzwischen in manchen Ländern gesetzlich vorgeschrieben, um eine Störung des Geschäftsbetrieb möglichst zu vermeiden. Eine Klassifizierung und Bewertung der Meldungen lässt sich nur abhängig vom System, der Verfügbarkeits- und Sicherheitsstufe vornehmen. Ein Patentrezept hierfür gibt es nicht. (sun)

LUKAS GRUNWALD

arbeitet als Consultant bei der DN Systems GmbH in Hildesheim und ist in diverse freie Softwareprojekte involviert.