Sicheres Logging mit Rsyslog

Geschütztes Logbuch

Kein Administrator kommt bei der Fehlersuche ohne gute Logfiles aus. Doch das beste Logging nützt nichts, wenn gerade die wichtigen Daten fehlen. Rsyslog sorgt für Zuverlässigkeit und Sicherheit. Kurt Seifried

Logfiles sind nur von eingeschränktem Nutzen, wenn ein Einbrecher sie auf dem Server verändern kann, um seine Spuren zu verwischen oder falsche Fährten zu legen. Wenn er faul ist, kann er sich auch damit begnügen, sie zu löschen, das war's dann mit der Spurensuche.

Für dieses alte Problem gibt es eine probate Lösung: Einfach die Logeinträge an einen anderen Rechner im Netz schicken, der gut auf die Dateien aufpasst. Auf dem Logging-Server lässt sich mit »syslog -r« der Syslog-Daemon so konfigurieren, dass er Logdaten von anderen Rechnern entgegennimmt. Fertig.

Doch leider ist diese Lösung gar nicht so toll, wie man denken könnte. So verwendet Syslog das UDP-Protokoll und UDP garantiert nicht die Zustellung der Daten. Wenn zum Beispiel eine Firewall aus Versehen den Syslog-Traffic blockiert, kriegen Sie davon gar nichts mit. Außerdem gibt es bei UDP keine Garantie, dass nicht jemand gefälschte Logdateien an den Logging-Server schickt.

Das geht zwar auch mit TCP, aber bei UDP um einiges einfacher, denn es gibt keinen Three-Way-Handshake und keine Sequenznummern, die man fälschen muss. Wenn ein Hacker wirklich auf Zack ist, kann er sogar die echten Nachrichten auf ihrem Weg zum Server verändern, ohne dass es jemand merkt.

Eine Lösung besteht darin, statt des normalen Syslog das Rsyslog-Paket zu verwenden. Auf Distributionen wie Debian, Ubuntu und Fedora ist es sogar schon der Standard-Logger, aber leider nicht immer auf dem neuesten Stand. Zum Beispiel bringt Ubuntu 10.04 LTS noch Rsyslog 4.2.0 mit, bei Redaktionsschluss war die neueste Version aber bereits 5.6.0. Wer nicht die neuesten Features braucht, kann aber ruhig mit der beigelegten Version arbeiten.

Wer Rsyslog selbst kompiliert, sollte darauf achten, es mit MySQL- und TLS-Verschlüsselungs-Support zu konfigurieren:

```
./configure --enable-gnutls --enable-➥
mysql
```

Dazu sind selbstverständlich die Entwicklungspakete von MySQL und GNU-TLS nötig. Die anschließende Installation des Pakets ist etwas schwieriger, denn je nach Distribution muss man zuerst die Syslog-Pakete entfernen und dabei die auftretenden Abhängigkeitsprobleme umgehen. Bei künftigen Updates des Pakets wiederholt sich das Problem. Versierte Administratoren sollte deshalb aus dem Quellcode lieber ein richtiges Distributionspaket bauen, das die geforderten Abhängigkeiten erfüllt.

Zuverlässig

Über den Transport der Logdaten per TCP hinaus bietet Rsyslog noch die Verifizierung auf Anwendungsebene, sodass die Zustellung garantiert ist, selbst wenn auf der TCP-Ebene etwas schiefgehen sollte. Dazu verwendet es das RELP-Protokoll (Reliable Event Logging Protocol), das sich auf dem Client zum Beispiel in der Konfigurationsdatei »/etc/rsyslog.conf« folgendermaßen einstellen lässt:

```
*.*:omrelp:10.1.2.3:2514
```

So ist die zuverlässige Zustellung der Logdaten sichergestellt, zugleich aber

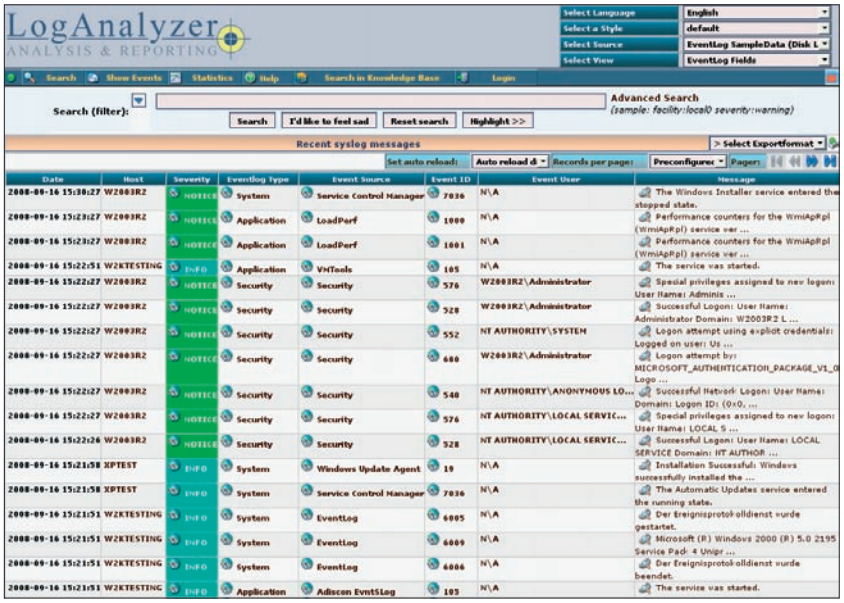

Abbildung 1: Loganalyzer bereitet die von Rsyslog gesammelten Daten übersichtlich auf.

auch aufgrund der IP-Adresse, dass der Absender einer Nachricht korrekt identifiziert wird, auch wenn er beispielsweise hinter einer NAT-Firewall steht. Nur so kann der Logging-Server die empfangenen Nachrichten auch nach einzelnen Hosts filtern.

Wie oben schon angedeutet ist es absolut unerlässlich, für eine sichere Übertragung aller Logdaten zu sorgen. Das soll nicht nur verhindern, dass ein Angreifer die Daten verfälscht, sondern auch, dass Unberechtigte sie überhaupt mitlesen können. Wenn zum Beispiel ein Benutzer beim SSH-Login statt seines Usernamens sein Passwort tippt, taucht es ungewollt auch im entsprechenden Log auf. Um jedes Abhören zu verhindern, verwendet Rsyslog die TLS-Verschlüsselung (Transport Layer Security, [3]). Wie man die benötigten Zertifikate mit dem »certtool« von GNU-TLS erzeugt und signiert, verrät die Anleitung.

Leider sind die Beispiele in der Rsyslog-Dokumentation nicht besonders nützlich, so verwendet die Konfiguration auf der TLS-Seite für »$InputTCPServerStreamDriverAuthMode« den Wert »anon«. Das bedeutet, dass keine Client-Authentifizierung stattfindet, womit auch nicht überprüfte Clients ihre Logdaten an den Server schicken können.

Wer in der Dokumentation für den GTLS Network Stream Driver nachsieht, findet dort die bessere Lösung »x509/name« [4]. Damit überprüft der Server zuerst das Zertifikat und den Namen, bevor er Daten vom Client entgegennimmt. Das Gleiche sollte man auf den Clients konfigurieren, um zu verhindern, dass ein Angreifer sich für den Server ausgeben kann. Die X509/Name-Konfiguration ist ab Rsyslog-Version 3.19.4 verfügbar.

Entlastung

Ein Problem bringt Remote-Logging mit sich: Es erzeugt eine dauerhafte Netzlast, die in einigen Fällen ganz erheblich sein kann, wenn zum Beispiel auf einmal viele Fehler auftreten. Wer Außenstellen mit wenig Bandbreite angebunden hat, kann eine Entlastung des Netzwerks durch etwas weniger Zuverlässigkeit erreichen: Rsyslog bietet mit Off-Peak Message Delivery ein Feature, bei dem es nicht kontinuierlich, sondern nur zu bestimmten Zeiten Logdaten an den Server schickt, Listing 1 zeigt ein Beispiel.

Damit schickt Rsyslog die Daten nur zwischen zehn Uhr abends und vier Uhr morgens an den Logserver. Dazwischen speichert es sie lokal zwischen. Wichtig ist die Option »$ActionQueueSaveOnShutdown«, die festlegt, dass Rsyslog beim Herunterfahren des Rechners die Daten auf der Festplatte speichert. Andernfalls sind die nur im Speicher gehaltenen Daten verloren.

Ein weiterer Vorteil der zeitgesteuerten Zustellung ist, dass sich so das Logging mehrerer Clients auf unterschiedliche Zeiten verteilen lässt, um eine Überlastung des Servers zu vermeiden [5].

Wer Rsyslog verwenden möchte, aber noch alte Unix-Systeme betreibt, die er nicht upgraden kann, muss nicht verzweifeln. Rsyslog unterstützt über eine Konfigurationsoption zur Not auch noch das Syslog-Protokoll über UDP:

```
@ModLoad imudp
$InputUDPServerRun 514
```

Weil Rsyslog so modular aufgebaut ist, unterstützt es auch mehrere Möglichkeiten der Ein- und Ausgabe wie UDP, TCP, RELP und so weiter. Auch für Systeme, die sich wirklich nicht auf Rsyslog upgraden lassen, gibt es eine Lösung: Man kann Stunnel verwenden, um die Nachrichten in SSL zu kapseln. Das funktioniert mit Syslog ebenso wie mit anderen in Stunnel verpacken Diensten.

Stunnel setzt dabei auf dem Server so auf, dass es Verbindungen annimmt, aber auf dem Client so, dass es Daten an den Server schickt, die es auf einem lokalen Port entgegennimmt. Der Syslog-Daemon auf dem Client wird so konfiguriert, dass er auf eben diesem Port loggt. Zur Visualisierung der gesammelten Rsyslog-Daten gibt es zahlreiche Pakete, zum Beispiel PHP Logcon oder Loganalyzer (Abbildung 1). *(ofr)* ■

Infos

[1] Rsyslog: [http://www.rsyslog.com]

[2] RELP: [http://www.librelp.com/relp.html]

[3] Encrypting Syslog Traffic with TLS (SSL): [http://www.rsyslog.com/doc-rsyslog_tls.html]

[4] GTLS Network Stream Driver: [http://www.rsyslog.com/doc/ns_gtls.html]

[5] Zeitgesteuerte Zustellung: [http://wiki.rsyslog.com/index.php/OffPeakHours]

Listing 1: Off-Peak Message Delivery

```
01 # reliably transmit messages
02 # during off-peak hours (10p to 4a)
03 $ModLoad omrelp
04 $WorkDirectory /rsyslog/work # where to place the
   spool files?
05 $ActionQueueType LinkedList
06 $ActionQueueDequeueTimeBegin 22
07 $ActionQueueDequeueTimeEnd 4
08 $ActionQueueFileName relpact
09 $ActionQueueSaveOnShutdown on
10 *.* :omrelp:10.1.2.3:2514
```