Syslog nach RFC

Martin Schütte

BSD Syslog

Design

- ursprünglich als Programmier-/Debug-Hilfe
- einfach zu benutzen und zu konfigurieren
- minimale Ressourcen, *best effort* Ansatz
- mit UDP auch auf andere Rechner loggen
- de facto Standard für Log-Daten unter Unix

BSD Syslog

Bestandteile

Kombination aus:

- API
- Nachrichtenformat
- Daemon
- IPC Protokoll

API

```
SYSLOG(3)              NetBSD Library Functions Manual       SYSLOG(3)

NAME
     syslog, vsyslog, openlog, closelog, setlogmask -- control system log

LIBRARY
     Standard C Library (libc, -lc)

SYNOPSIS
     #include <syslog.h>

     void
     syslog(int priority, const char *message, ...);

     void
     openlog(const char *ident, int logopt, int facility);

     void
     closelog(void);

     int
     setlogmask(int maskpri);
```

Syslog Message Format

```
<38>Mar 17 21:57:57 frodo sshd[701]: Connection from 211.74.5.81 port 5991
<52>Mar 17 13:54:30 192.168.0.42 printer: paper out
```

- Priority
- Header
  - Timestamp
  - Hostname
- Message
  - Tag
  - Content

Konventionen:

- Priority nicht in Logdatei
- nur ASCII
- bis zu 1024 Zeichen pro Zeile

Daemon: syslogd

- Liest Nachrichten von Kernel, Anwendungen, Netzwerk
- Filter nach Priority
- schreibt in Dateien, Pipes, auf Terminals oder Netzwerk (UDP)
- neue Implementierungen mit mehr Features
  (z. Bsp. *BSD syslogd, syslog-ng, rsyslog)
  - Filter nach Host-/Programmname oder RegExp
  - Konvertieren verschiedener Nachrichten und Zeitstempel
  - Schreiben in Speicherpuffer, TCP-Verbindungen oder SQL-DBs

“Transport Protocol“

Nachrichten von . . .

- Anwendungen: `socket(AF_UNIX, SOCK_DGRAM, 0);`
- Netzwerk: `socket(AF_INET, SOCK_DGRAM, 0);`
- Kernel: `open("/dev/klog", O_RDONLY, 0);` (Ringspeicher)

⇒ Eine Nachricht/Zeile je `recvfrom()`/`read()`.

IETF Working Group

"Security Issues in Network Event Logging"

- RFC 3164: The BSD Syslog Protocol (informational)
- RFC 3195: Reliable Delivery for Syslog
- RFC 5424: The Syslog Protocol
- RFC 5425: TLS Transport Mapping for Syslog
- RFC 5426: Transmission of Syslog Messages over UDP
- Internet Drafts:
  - Signed Syslog Messages
  - Syslog Management Information Base

RFC 5424: The Syslog Protocol

Neues Nachrichtenformat

- Plain Text
- ISO Zeitstempel (mit Jahr, Zeitzone und hoher Auflösung)
- FQDNs
- Message IDs (vgl. Windows Eventlog)
- Strukturierte Daten mit Namensräumen (über SNMP Private Enterprise ID)
- UTF-8 für Daten und Nachrichtentext
- mindestens unterstützte Nachrichtenlängen: 480 octets MUST, 2048 octets SHOULD

Syslog-protocol Nachrichtenformat

```
<165>1 2003-10-11T22:14:15.003Z frodo.example.com prog - ID47↩
 [exampleSDID@17660 iut="3" eventID="1011" eventSource="Application"]↩
 BOMUne entré du journal des événements ...
```

- Header
  - Priority
  - Version *(new)*
  - Timestamp *(extended)*
  - Hostname
  - Application Name
  - Process ID
  - Message ID *(new)*
- Structured Data *(new)*
- Message text *(UTF-8)*

Änderungen an `syslog(3)`

bisher nur in NetBSD-current

- API ungeändert
- Syslog-Protocol in IPC zwischen `syslog(3)` und `syslogd(8)`:
  - ISO Zeitstempel
  - FQDN
  - neue Felder (MSGID und SD) leer

Neuer API-Vorschlag: `syslogp(3)`

bisher nur in NetBSD-current

- minimale Erweiterung für MSGID und SD

≈ `syslog(3)` mit drei Nachrichten-Parametern

```
void syslogp(int priority, const char *msgid,↩
    const char *sdfmt, const char *message, ...);
void vsyslogp(int priority, const char *msgid,↩
    const char *sdfmt, const char *message, va_list args);

syslog(LOG_INFO, "foobar error: %m");
syslogp(LOG_INFO, NULL, NULL, "foobar error: %m");

syslogp(LOG_INFO, "ID%d", "[meta language=\"de-DE\"]",↩
    "Ereignis: %s", 42, EventDescription);
```

RFC 5426: UDP Transport

- UDP-Übertragung wie bisher
- Portnummer: udp/514
- eine Nachricht pro UDP-Datagramm
- keinerlei Sicherung/Authentifizierung
- akzeptierte Nachrichtenlängen: mindestens
  480 octets (MUST bei IPv4) bzw. 2048 octets (SHOULD)

RFC 5425: TLS Transport

- Punkt-zu-Punkt Verschlüsselung, Integrität und Authentifizierung
- Portnummer: tcp/6514
- Erfordert Client- und Serverzertifikate
- Authentifizierung per CA oder Zertifikat/Subject/Fingerprint
- Format: Nachrichtenlänge, Leerzeichen, Nachricht
- akzeptierte Nachrichtenlängen: mindestens 2048 octets (MUST) bzw. 8192 octets (SHOULD)

syslog.conf Beispiel

```
tls_ca="/etc/my.cacert"
tls_cert="/etc/localhost.crt"
tls_key="/etc/localhost.key"
tls_gen_cert=on

tls_server=on
tls_bindhost="192.168.1.2"
tls_bindport="syslog-tls"
tls_allow_clientcerts="/etc/somehost.crt"

*.*     @[fe80::211:9ff:fe41:be53]:syslog-tls(↩
                fingerprint="MD5:00:A2:...:27")
```

Internet-Draft: Signed syslog Messages

Übersicht

- digital signierte Syslog-Nachrichten
- Ende-zu-Ende Authentifizierung, Integrität, korrekte Reihenfolge und Vollständigkeit
- DSA für Signaturen
- SHA-1 oder SHA-256 als Hashfunktion
- OpenPGP- oder X.509-Zertifikate

Internet-Draft: Signed syslog Messages

Konstruktion

- Signature Groups für mehrere Nachrichtenströme
- Voraussenden des öffentlichen Schlüssels
- Übertragen der Folge von Nachrichten in Folge von Hash-Werten
- Einfügen der Signaturnachrichten in den Nachrichtenstrom
- Signieren der eigenen Kontrollnachrichten

Payload Blocks

Öffentlichen Schlüssel versenden

```
<110>1 2008-08-02T01:09:27.773505+02:00 host.example.org syslogd - - ↩
 [ssign-cert VER="0111" RSID="1217632162" SG="3" SPRI="0" TBPL="1059"↩
 INDEX="1" FLEN="1059" FRAG="2008-08-02T01:09:27.773464+02:00 C↩
 MIIC+jCCArmgAwIBAwIBA...YA==" SIGN="MC0CFFEHx8UX32vEW...k+o="]
```

Beim Systemstart, enthält:

- Signature Group (`VER`, `RSID`, `SG`, `SPRI`)
- Fragmentierung (`TBPL`, `INDEX`, `FLEN`)
- Payload Block (`FRAG`) mit
  - Zeitstempel
  - Art des Schlüssels
  - öffentlicher Schlüssel (base64)
- DSA Signatur (`SIGN`)

Signature Blocks

SHA-1 Hash-Werte sammeln ...

```
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg0
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg1
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg2
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg3
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg4
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg5
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg6
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg7
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg8
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg9
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg10
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg11
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg12
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg13
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg14
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg15
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg16
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - msg17

siUJM358eYFHOS2KOMTlveWeH/U= zTxfthW8WqmtFhOG4k/+ZxkirTA= j9dubU1GNVp7qWShwph/w32nDO8=
 XQDLZ/NuwirmLdMORtm84r9kIW4= RNDFNCo7hiCsK/EKumsPBbFHNZA= ANiE3KbY948J6cEB640fAtWXuO4=
 e2M/OqjHDfxLVUSPt1CsNJHm9wU= Y+racQst7F1gR8eEUh8O7o+M53s= JAMULRxjMPbOO5EhhKbsUkAwbl0=
 pd+N5kmlnyQOBoItELd/KWQrcMg= dsMQSzPHIS6S3Vaa23/t7U8JAJ4= i4rE3x7N4qyQGTkmaWHsWDFP9SY=
 qgTqV4EgfUFd3uZXNPvJ25erzBI= XWOYrME5kQEh+fxhg1fetnWxfIc= 7YPcRHsDwXWnQuGRWaJtFWw9hus=
 PIvLmOmh+he5+PDihG1p7sQlx8k= lPzUvxOI1VwSGWV7yKF9W//Yb2U= X+PWYcx5AXnsDVSNAHLZUGk5ioY=
```

Signature Blocks

. . . und Hash-Werte & Signatur versenden

```
<110>1 2008-08-02T01:09:32.399406+02:00 host.example.org syslogd - - [ssign VER="0111"
 RSID="1217632162" SG="3" SPRI="0" GBC="4" FMN="1" CNT="20" HB="siUJM358eYFHOS2KOMTlveWeH/U=
 zTxfthW8WqmtFhOG4k/+ZxkirTA= j9dubU1GNVp7qWShwph/w32nDO8= XQDLZ/NuwirmLdMORtm84r9kIW4=
 RNDFNCo7hiCsK/EKumsPBbFHNZA= ANiE3KbY948J6cEB640fAtWXuO4= e2M/OqjHDfxLVUSPt1CsNJHm9wU=
 Y+racQst7F1gR8eEUh8O7o+M53s= JAMULRxjMPbOO5EhhKbsUkAwbl0= pd+N5kmlnyQOBoItELd/KWQrcMg=
 dsMQSzPHIS6S3Vaa23/t7U8JAJ4= i4rE3x7N4qyQGTkmaWHsWDFP9SY= qgTqV4EgfUFd3uZXNPvJ25erzBI=
 XWOYrME5kQEh+fxhg1fetnWxfIc= 7YPcRHsDwXWnQuGRWaJtFWw9hus= PIvLmOmh+he5+PDihG1p7sQlx8k=
 lPzUvxOI1VwSGWV7yKF9W//Yb2U= X+PWYcx5AXnsDVSNAHLZUGk5ioY= okXY88MGG4QybrYMf8HJN23WO1Y=
 HcaPyHfQ2s1SuSciTKw4woYWuMg=" SIGN="MCwCFFrOi6taT1vWowR7yc5bEQxFfY7/Ah...IQ=="]
```

Eine Syslog-Nachricht, enthält:

- Signature Group (`VER`, `RSID`, `SG`, `SPRI`)
- Global Block Counter (`GBC`, zählt syslog-sign Nachrichten)
- First Message Number (`FMN`, zählt normale Nachrichten)
- `CNT` Hash Blocks (`HB`)
- DSA Signatur (`SIGN`)

Offline Verifikation

1. Log teilen in Certificate Blocks (CB), Signature Blocks (SB) und normale Nachrichten
2. für normale Nachrichten Hashes berechnen
3. CBs und SBs sortieren
4. CBs und SBs verifizieren und zusammensetzen
   $\Rightarrow$ ergibt öffentliche Signaturschlüssel
5. SBs verifizieren
6. Folge aller Hash-Werte aufbauen (mittels `FMN`)
7. zu jedem Hash-Wert die zugehörige Nachricht raussuchen
   $\Rightarrow$ ergibt Folge verifizierter Nachrichten

verify.pl

```
$ perl verify.pl -i test.log
reading input...
processing CBs...
decoding SGs...
got PKIX DSA key
verifying CBs...
verified CB and got key for SG: (host.example.org,1217632162,0111,3,0),↩
    start: 2008-08-02T01:09:27.773464+02:00
now process SBs
signed messages:
...
host.example.org,1217632162,0111,3,0,11 <15>1 ... test 6255 - - msg10
host.example.org,1217632162,0111,3,0,12 <15>1 ... test 6255 - - msg11
host.example.org,1217632162,0111,3,0,13 **** msg lost
host.example.org,1217632162,0111,3,0,14 <15>1 ... test 6255 - - msg13
host.example.org,1217632162,0111,3,0,15 <15>1 ... test 6255 - - msg14
host.example.org,1217632162,0111,3,0,16 <15>1 ... test 6255 - - msg15
...

messages without signature:
<15>1 2008-08-02T02:09:27+02:00 host.example.org test 6255 - - modified msg12
```

Implementierungen

- rsyslog (ab Version 3.19)
- syslog-ng (in OSE ab Version 3.0)
- NetBSD syslogd (in -current)

Umstieg auf neue Protokolle

- syslogd versteht altes und neues Format
- Parallel zwei syslogd-Instanzen sind nicht praktikabel
- CA erleichtert die TLS-Konfiguration sehr
- Hauptproblem: Log-Analyseprogramme
  (müssen neues Format parsen können)

ToDo-Liste

- Implementierungen testen
- Syslog-Sign RFC fertigstellen
- Log-Infrastruktur upgraden
- Structured Data Feld benutzen
- einheitliche API finden (`syslogp(3)`)

Links

- IETF Syslog WG: Status Page
- rsyslog
- syslog-ng
- NetBSD & GSoC: syslogd
- NetBSD CVS Repositories