

Syslog

Martin Schütte

25. März 2007



Gliederung

Einführung

- Sinn und Ziel von Logfiles

BSD Syslog

- Konzept

- Nachrichten senden

- Server

- Loggen im Netzwerk

- Storage

neue Protokolle

Analyse

- Klassifizierung

- Zeitnahe Verfolgung

Sinn von Logfiles

- Fehlersuche
- Systemauslastung
- Protokollierung/Zurechenbarkeit von Aktivitäten
- Warnung vor laufenden Angriffen
- Untersuchung nach Angriffen

Was und wieviel loggen?

Mindestens loggen?

- Änderungen an Konfiguration oder Diensten
- Privilegien-Wechsel (login, su, sudo, ...)
- Umgehungsversuche bei Sicherheitsmechanismen
- Zugriff auf vertrauliche Daten
- immer: *genaue Uhrzeit!*

Was sonst alles?

- lieber mehr als weniger
- Beschränkung durch Plattenplatz, Bandbreite, Analysemöglichkeit
- genug Reservern für Spitzenlast berücksichtigen

Wie lange speichern?

Systemlogs unbegrenzt aufbewahren.

aber:

Viele Logeinträge sind auch personenbezogene Daten
(E-Mail, FTP-Transfers, Weblogs, Shell-Logins, ...)

Möglichkeiten:

- garnicht erst erfassen
- analysieren, aber nicht speichern
- nach n Tagen löschen, nicht archivieren
- Bei Zugriff mehrerer Personen auch denkbar:
 - Pseudonymisieren

Beschränkungen von Syslog/Logfiles

- Logging meist optional, immer nur *best effort*
- Nachrichten können verloren gehen
- Reihenfolge der Nachrichten unsicher
- Lokale Dateien generell manipulierbar

Für hohe Sicherheitsanforderungen: Audit-Systeme

- z. B. Protokollieren jedes System-Calls
- erst Protokoll schreiben, dann Aktion ausführen
- i. d. R. nur lokal möglich

BSD Syslog

- ursprünglich von Eric Allman für sendmail geschrieben
- de facto Standard für Log-Files unter Unix
- Systemweit einheitlicher Log-Mechanismus
- Metadaten über Teilsystem und Dringlichkeit
- Loggen in Datei, auf Konsole oder per UDP/IP ins Netz

Metadata: Facility & Priority

Facility:

0. kernel messages
1. user-level messages
2. mail system
3. system daemons
4. security/authorization messages
5. messages generated internally by syslogd
6. line printer subsystem
7. network news subsystem
8. UUCP subsystem
9. clock daemon
10. security/authorization messages
11. FTP daemon
12. NTP subsystem
13. log audit
14. log alert
15. cron daemon
16. local use 0 (local0)
17. local use 1 (local1)
18. local use 2 (local2)
19. local use 3 (local3)
20. local use 4 (local4)
21. local use 5 (local5)
22. local use 6 (local6)
23. local use 7 (local7)

Priority:

0. Emergency: system is unusable
1. Alert: action must be taken immediately
2. Critical: critical conditions
3. Error: error conditions
4. Warning: warning conditions
5. Notice: normal but significant condition
6. Informational: informational messages
7. Debug: debug-level messages

Aufbau einer Syslog-Zeile

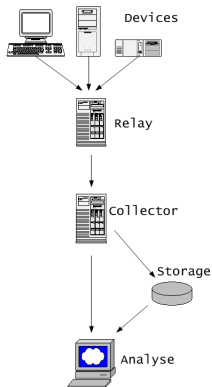
```
<38> Mar 17 21:57:57 frodo sshd[27010]: Connection from 211.174.53.81 port 55991
```

```
<52> Mar 17 13:54:30 192.168.0.42 printer: paper out
```

- PRI: Facility*8 + Priority
- Header
 - Timestamp
 - Hostname
- Message
 - Tag
 - Content

Message-Inhalt völlig beliebig, Zeichencodierung und minimale/maximale Zeilenlänge nicht festgelegt („sicher“: nur ASCII, bis zu 1023 Zeichen)

Rollenverteilung und Datenfluss



Syslog-Rollen:

- *Devices/Sender* erzeugen Meldungen
- *Relays* empfangen und leiten weiter
- *Collectors* sammeln Meldungen

Außerdem:

- *Storage* zur Archivierung
- *Analyzer* zum Auswerten der Meldungen

Nachrichten senden

- In Programmen:

```
#include <syslog.h>
openlog("my_tool", LOG_PID, 0);
setlogmask(LOG_UPTO(LOG_INFO));
syslog(LOG_LPR|LOG_ALERT, "printer on fire");
closelog();
```

- Von der Shell:

```
/usr/bin/logger -t my_tool -i -p lpr.alert \  
"printer on fire"
```

- oder per Pipe (z. B. in httpd.conf):

```
ErrorLog "|/usr/bin/logger -p local3.info -t apache"
```

Übergabe an Server

- syslog(3) sendet Nachrichten an lokalen Socket:
/var/run/log oder /dev/log
- Kernel-Nachrichten über Spezialdatei:
/dev/klog oder /proc/kmsg
- im Netz per UDP/IP, Port 514

syslogd

- meist der vorkonfigurierte Syslog-Daemon
- Optionen und Parameter systemabhängig
- Kommuniziert über lokale Sockets und mit UDP im Netz
- Filtert Nachrichten anhand der Facility & Priority

syslog.conf (1)

Jede Regel hat das Format: Selektor <Tab> Aktion

Selektoren:

- facility.level
- facility1,facility2.level
- facility1.level1;facility2.level2
- *.level
- *.level;badfacility.none

Aktionen:

- filename in Datei schreiben
- -filename gepuffert in Datei schreiben
- @host an Host weiterleiten
- user1,user2,... Usern aufs Terminal schreiben
- * allen Usern aufs Terminal schreiben

syslog.conf (2)

Systemabhängige Erweiterungen:

- level-Selektion mit =, =>, =<, !
- Auswahl nach Rechner (+host) oder Programm (!tag)
- Meldungen an Programm oder Pipe übergeben
(|command oder |/some/fifo)

Beispiel:

```
*.info                @192.168.5.23

security,auth.*       /var/log/security
mail.!=debug          /var/log/maillog

*.=debug              /var/log/debug
*.*!=debug;mail.none /var/log/messages
```

syslog-ng

- Alternativer syslogd von Balázs Scheidler
- verschiedene Quellen
- Filter für Nachrichteninhalte
- verschiedene Datumsformate, Zeitzonen, Sende-/Empfangszeiten, Millisekunden in Zeitstempel
- Makros für Logdateien über Makros wählen
- Syslog über TCP
- basiert grundsätzlich auf Logpfaden:

```
log { source(s1); source(s2);  
      filter(f1); filter(f2);  
      destination(d1); destination(d2);  
      flags(flag1); };
```

syslog-ng.conf (1)

Sources:

- `internal()`; syslog-ng selbst
- `file("/dev/kmsg");` Kernel-Nachrichten (kein `tail -f`)
- `pipe("/var/tmp/mypipe");` benannte Pipes
- `unix-dgram("/var/run/log");` `unix-stream()`; Sockets
- `tcp(ip(192.168.1.2) port(1234));` `udp()`; Netzwerk

Destinations:

- `file("/var/log/$HOST/$YEAR$MONTH/messages");` In Textdatei
- `pipe("/var/tmp/mypipe");` benannte Pipes
- `unix-dgram("/var/tmp/socket");` Sockets
- `tcp("192.168.2.3" port(514));` Netzwerk
- `usertty(root)` Nachricht an User-Konsole senden
- `program("/bin/logsurfer");` Nachrichten an ein Programm senden

syslog-ng.conf (2)

Filters:

- `facility(mail);`
- `priority(crit..emerg);`
- `host(frodo);` Regex für Hostnamen
- `netmask(192.168.4.0/255.255.255.0);` IP eines Clients
- `program(postfix/.*);` Regex für Tag
- `match(mail);` Regex für Content
- `filter(mein_filter);` Auswerten eines anderen Filters

Flags:

- `final` Nachricht nicht weiterverarbeiten
- `fallback` Nur Nachrichten annehmen, die in keinem andern Pfad matchen
- `catchall` Nachrichten aus allen Quellen verarbeiten
- `flow-control` Blockieren falls die Destination nicht liest

syslog-ng.conf (3)

Templates: eigene Ausgabeformate

```
template t_iso      { template("$R_ISODATE $HOST $MSG\n");
                    template_escape(no); };
destination d_iso  { file("/var/log/$YEAR/$MONTH/messages"
                    template(t_iso) frac_digits(3)) ;};

template t_sql     { template("INSERT INTO log VALUES
                            ('$YEAR-$MONTH-$DAY', '$hour:$min:$sec',
                            '$MSG');\n"); template_escape(yes); };
destination d_sql  { pipe("/tmp/mysql.pipe" template(t_sql)); ;}
```

Ergebnis:

```
2007-02-25T18:27:24.637+01:00 fax syslog-ng[52472]: \
  syslog-ng starting up; version='2.0.0'

INSERT INTO log VALUES ('2007-02-25', '18:27:24', \
  'syslog-ng[52472]: syslog-ng starting up; version=\`2.0.0\` ');
```

syslog-ng.conf (4)

```
source src { unix-dgram("/var/run/log"); internal(); };
source remote { udp(); tcp(port(514)); };

destination messages { file("/var/log/messages"); };
destination maillog { file("/var/log/maillog"); };
destination loghost { udp("192.168.5.23" port(514)); };

filter f_security { facility(security) or facility(auth); };
filter f_mail { facility(mail); };
filter f_info { level(info..emerg); };

log { source(src); filter(f_info); destination(loghost); };

log { source(src); filter(f_mail); destination(maillog);
      flags(final); };

log { source(src); source(remote); destination(messages); };
```

BSD Syslog über UDP

Nachteile:

- Paketverlust
- Absenderidentität unsicher
- Fälschen von Paketen
- Überfluten mit falschen Paketen

Vorteile (?):

- einfach und effizient
- „Stealth Logging“ möglich

BSD Syslog über TCP

Nachteile:

- verschiedene Implementierungen – ggf. nicht untereinander kompatibel
- etwas ineffizienter

Vorteile:

- zuverlässige Verbindung
- lässt sich mit SSH oder SSL absichern

BSD Syslog über TLS

- Daemon benutzen, der über TCP oder Pipe sendet
- TCP-Verbindung über SSH oder SSL/TLS tunneln
- Programme: z. B. ssh, SSL-netcat, stunnel

stunnel.conf

```
# globale Optionen:  
chroot = /var/tmp/stunnel  
pid = /stunnel.pid  
setuid = stunnel  
setgid = stunnel  
  
# SSL-Zertifikate und -Schlüssel:  
cert = /usr/local/etc/my.crt  
key = /usr/local/etc/my.key  
CAfile = /usr/local/etc/CA.crt  
verify = 2  
  
# ein Service:  
[syslog-ssl]  
client = yes  
accept = 127.0.0.1:1514  
connect = 192.168.5.23:514
```

Logfiles rotieren

Problem:

- Logs sollen lange aufbewahrt werden
- Logdateien sollen nicht unbeschränkt wachsen

Notlösung:

```
#!/bin/sh
cd /var/log
mv logfile.2.bz2 logfile.3.bz2
mv logfile.1.bz2 logfile.2.bz2
mv logfile logfile.1
cat /dev/null > logfile
kill -HUP `cat /var/run/syslogd.pid`
bzip2 logfile.1
```

Programme: unter BSD newsyslog, unter Linux logrotate

Beispielkonfiguration

newsyslog.conf:

```
# logfilename      [owner:group] mode count size  when  flags [/pid_file] [sig_num]
/var/log/messages  600   50  2048 $W5D0   J0
/var/log/apache/*.log www:www 644   2    * $M1D0   GB /var/run/httpd.pid 30
```

/etc/logrotate.conf oder /etc/logrotate.d/syslog:

```
errors sysadmin@example.com
compress
compresscmd /usr/bin/bzip2

/var/log/messages /var/log/localmessages {
    weekly
    rotate 50
    size 2M
    create 640 root root
    postrotate
        /etc/init.d/syslog reload
    endscrip
}
```

Besser: Rotation vermeiden

- mit syslog-ng: wöchentlich neue Datei schreiben

```
destination archiv { file("/var/log/$YEAR/$MONTH/$WEEK/file")
```

- mit cronolog:

```
| cronolog /var/log/\%Y/\%m/\%W/file
```

- per cronjob dann alte Logs komprimieren oder löschen:

```
tar cjf /var/log/'date +%Y-%m-%W'.tar.bz2 \  
/var/log/'date +%Y/%m/%W'/messages
```

IETF syslog-sec Working Group

- RFC 3164: The BSD Syslog Protocol (informational)
- RFC 3195: Reliable Delivery for Syslog
- aktuelle Drafts:
 - The Syslog Protocol
 - Transmission of Syslog messages over UDP
 - TLS Transport Mapping for Syslog
 - Signed Syslog Messages
 - Syslog Management Information Base
- weitere Vorschläge:
 - DTLS transport mapping for Syslog
 - SSH transport mapping for Syslog

RFC 3195: Reliable Delivery for Syslog

- BEEP als Transportprotokoll
- XML-Datensätze
- RAW-Mode ähnlich wie BSD Syslog
- COOKED-Mode:

```
C: MSG 2 3 . 4360 250
C: Content-Type: application/beep+xml
C:
C: <entry facility='24' severity='5'
C:   timestamp='Oct 27 13:24:12'
C:   deviceFQDN='screen.lowry.records.example.com'
C:   deviceIP='10.0.0.47'
C:   pathID='173'
C:   tag='dvd'>
C:     Job paused - Boss watching.
C: </entry>
C: END
```

```
S: RPY 2 3 . 1383 45
S: Content-Type: application/beep+xml
S:
S: <ok/>
S: END
```

Draft: The Syslog Protocol

- weiterhin Textzeilen
- unabhängig vom Transport-Protokoll
- vollständiger Zeitstempel
- Message IDs
- strukturierte Felder in UTF-8 (eindeutig benannt und automatisch auszuwerten)

Aufbau einer Syslog-Zeile

nach syslog-protocol

- Header
 - PRI: Facility*8 + Priority
 - Version
 - Timestamp
 - Hostname
 - Application Name
 - Process ID
 - Message ID
- Structured Data (UTF-8)
 - ID
 - name=value pairs
- free-form Message

Aufbau einer Syslog-Zeile

nach syslog-protocol; Beispiel

<165>1 2003-10-11T22:14:15.003Z machine.example.com evntslog - ID47

[exampleSDID@0 iut="3" eventID="1011" eventSource="Application"]

BOMAn application event log entry ...

Analyse von Logfiles

99,9% der Logs sind unwichtig

```

Jul 9 09:37:58 localhost SystemStarter: Willkommen!
Jul 9 09:37:58 localhost lookupd[122]: lookupd (version 324.2.1) starting - Fri Jul 9 09:37:58 2004
Jul 9 09:37:58 localhost ConsoleMessage: Starting Apple Multicast DNS Responder
Jul 9 09:37:58 localhost ConsoleMessage: Starting kernel event agent
Jul 9 09:37:58 localhost ConsoleMessage: Starting timed execution services
Jul 9 09:37:58 localhost ConsoleMessage: Starting SecurityServer
Jul 9 09:38:01 localhost SystemStarter: ?Apple Multicast DNS Responder? wird gestartet
Jul 9 09:38:02 localhost SystemStarter: ?Kernel Event Agent? wird gestartet
Jul 9 09:38:02 localhost SystemStarter: ?Timed Execution Services? werden gestartet
Jul 9 09:38:04 localhost kernel: ApplePMUUserClient::setProperties WakeOnACchange 0
Jul 9 09:38:06 localhost ConsoleMessage: Initializing network
Jul 9 09:38:06 localhost mDNSResponder[155]: mDNSResponder-58.8 (Apr 24 2004 20:38:40) starting
Jul 9 09:38:06 localhost SystemStarter: Starting SecurityServer
Jul 9 09:38:06 localhost SystemStarter: Netzwerkdienste werden initialisiert
Jul 9 09:38:06 localhost ConsoleMessage: Checking disks
Jul 9 09:38:06 localhost SystemStarter: Volumes werden ?berpr?ft
Jul 9 09:38:08 localhost kernel: ATY_Bee_A: vram [9c000000:02000000]
Jul 9 09:38:08 localhost kernel: ATY_Bee_B: vram [98000000:02000000]
Jul 9 09:38:08 localhost syslogd: /dev/console: Input/output error
Jul 9 09:38:08 localhost init: kernel security level changed from 0 to 1
Jul 9 09:38:09 localhost DirectoryService[186]: Launched version 1.8 (v256.6)
Jul 9 09:38:09 localhost loginwindow[182]: Sent launch request message to DirectoryService mach_init port
Jul 9 09:38:16 localhost SystemStarter: Warten auf ?Netzwerkdienste werden initialisiert?
Jul 9 09:38:22 localhost last message repeated 2 times
Jul 9 09:38:23 localhost config[87]: executing /System/Library/SystemConfiguration/Kicker.bundle/Contents/Resources/set-hostname
Jul 9 09:38:24 localhost set-hostname[195]: setting hostname to ivich.local
Jul 9 09:38:25 localhost SystemStarter: Warten auf ?Netzwerkdienste werden initialisiert?
Jul 9 09:38:28 localhost ConsoleMessage: Loading Shared IP extension
Jul 9 09:38:28 localhost SystemStarter: Shared-IP-Erweiterung wird geladen
Jul 9 09:38:28 localhost /usr/libexec/crashreporterd: get_exception_ports() failed: (ipc/send) invalid destination port
Jul 9 09:38:28 localhost ConsoleMessage: Starting Apple File Service
Jul 9 09:38:28 localhost ConsoleMessage: Starting printing services
Jul 9 09:38:29 localhost SystemStarter: ?Apple File Services? werden gestartet
Jul 9 09:38:29 localhost SystemStarter: Die Druckerdienste werden gestartet
Jul 9 09:38:30 localhost ConsoleMessage: Loading IP Firewall extension
Jul 9 09:38:30 localhost SystemStarter: IP-Firewall-Erweiterung wird geladen
Jul 9 09:38:32 localhost kernel: IP packet filtering initialized, divert enabled, rule-based forwarding enabled, default to accept, logging disabled
Jul 9 09:38:32 localhost kernel: IPv6 packet filtering initialized, default to accept, logging disabled
Jul 9 09:38:32 localhost kernel: IP firewall loaded
Jul 9 09:38:32 localhost ConsoleMessage: Starting internet services
Jul 9 09:38:32 localhost SystemStarter: ?Internet Services? werden gestartet
Jul 9 09:38:32 localhost xinetd[257]: xinetd Version 2.3.11 started with libwrap options compiled in.
Jul 9 09:38:32 localhost xinetd[257]: Started working: 3 available services
Jul 9 09:38:33 localhost SystemStarter: Systemstart beendet.
Jul 9 09:38:44 localhost diskarbitrationd[88]: disk1s2 hfs ED021A74-9EEF-3AA5-9B54-5B703F7D0D71 ute [not mounted]
Jul 9 09:38:45 localhost diskarbitrationd[88]: disk1s2 hfs ED021A74-9EEF-3AA5-9B54-5B703F7D0D71 ute /Users/ute

```

grep

- `grep "wichtige Meldung" logdatei | less`
- `grep -v "unwichtige Meldung1" logdatei | \`
`grep -v "unwichtige Meldung2" | \`
`grep -v "unwichtige Meldung3" | less`
- ineffizient
- nur interaktiv, nicht automatisch
- kein Auslösen von Aktionen

grep und andere Shell-Tools

```
cat logfile \
| egrep '^.{15} .+ sshd\[.+\]: Invalid user .+ from .+$' \
| sed -e 's/^\.{16}\//' -e 's/ sshd\[.*\]: / sshd: /' \
-e 's/user .* from/user XY from/' \
| sort | uniq -c | sort -nr >> report
```

Ergebnis:

```
Summary: lines.login_failed_user
```

```
-----
451 neo sshd: Invalid user XY from 210.75.23.122
451 mail sshd: Invalid user XY from 210.75.23.122
209 neo sshd: Invalid user XY from 60.191.41.89
 12 neo sshd: Invalid user XY from 125.152.17.236
 12 mail sshd: Invalid user XY from 125.152.17.236
```

Grundprinzip: Artificial Ignorance

Ziel:

- Automatisches Filtern der uninteressanten Log-Daten
- schnell auf „ungewöhnliche“ Meldungen aufmerksam werden

Vorgehensweise:

1. vorhandene Logdaten auswerten
2. häufigste normale Meldungen mit Regex ausfiltern
3. das was übrigbleibt regelmäßig ansehen
4. Filter ggf. nachjustieren

logcheck

- Klassifiziert alle Zeilen mit Regex in Kategorien, z. B.:
 - hacking
 - violations
 - ignore
 - alles übrige
- schickt Report per E-Mail
- merkt sich letzte Position in der Logdatei – dadurch beliebig oft aufrufbar

Falls möglich: Spezialprogramme

Postfix log summaries for Jan 14

Grand Totals

messages

```

3328  received
6981  delivered
   1   forwarded
   67  deferred (854  deferrals)
   98  bounced
4828  rejected (40%)
   0   reject warnings
   0   held
   24  discarded (0%)

```

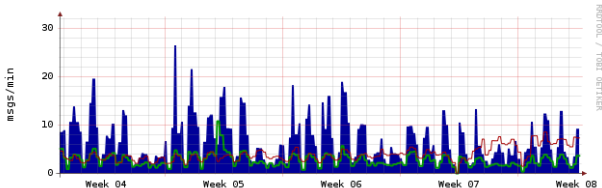
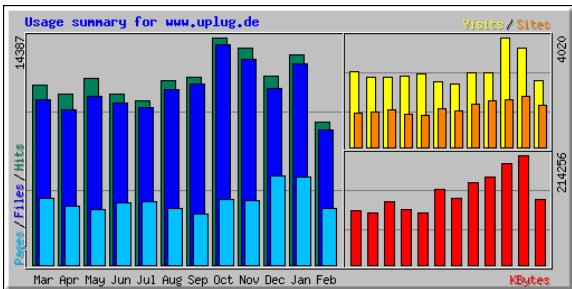
```

53388k bytes received
107479k bytes delivered

```

...

Ereignisse zählen



■ Sent	total:	263230 msgs	avg:	6.62 msgs/min	max:	3027 msgs/min
■ Received	total:	108341 msgs	avg:	2.72 msgs/min	max:	836 msgs/min
■ Rejected	total:	156771 msgs	avg:	3.95 msgs/min	max:	69 msgs/min

[Thu Feb 22 17:23:19 2007]

„Echtzeit“-Analyse

Vorteile:

- Abfolge und Abhängigkeiten der Nachrichten überwachen
- zustandsbasierte Auswertung (z. B. Sessions verfolgen)
- automatische Reaktionen ausführen (z. B. Benachrichtigungen oder Gegenmaßnahmen)

Programme:

- swatch
- logsurfer
- Simple Event Correlator

Swatch

- prüft zeilenweise nach Regex und führt Aktionen aus

swatch.conf Beispiel:

```
ignore /ntpd.*: kernel time sync enabled/
```

```
watchfor /\,'su root\' failed/
```

```
bell
```

```
exec echo $0 | mail -s\"security alert\" root@example.com
```

logsurfer

- prüft zeilenweise nach Regex und führt Aktionen aus
- dynamisches Erzeugen/Löschen von Regeln
- Sammeln mehrerer Zeilen in Kontexte
- komplizierte Konfigurationsdatei (durch mehrere Quoting-Ebenen)

logsurfer.conf Beispiel:

```
'on (/.*): file system full$' - - - 0 pipe "report.sh alarm"
',.*' - - - 0 pipe "/bin/cat"
```

Simple Event Correlator

- kann Ereignisse zählen und Zeitspannen beachten
- dynamisches Erzeugen/Löschen von Regeln
- Erzeugen beliebiger eigener Ereignisse
- Benutzen beliebiger Kontext-Namen
- Ausführen beliebiger Perl-Funktionen

Simple Event Correlator

Aktionen:

- `Single` – Sofort Aktion ausführen
- `SingleWithScript` – Skript ausführen, dann ggf. Aktion
- `SingleWithSuppress` – Sofort Aktion ausführen, Ereignis für t Sekunden ignorieren
- `Pair` – Sofort Aktion ausführen, Ereignis ignorieren bis zweites Ereignis eintritt, dann zweite Aktion ausführen
- `PairWithWindow` – wie `Pair` aber mit Timeout und zugehöriger Aktion
- `SingleWithThreshold` – Ereignisse in Zeitfenster zählen, falls n Ereignisse in t Sekunden, dann Aktion ausführen
- `SingleWith2Thresholds` – wie `SingleWithThreshold`, mit zweitem Zeitfenster und zugehöriger Aktion
- `Suppress` – Zeile ignorieren
- `Calendar` – zu festgelegter Zeit Aktion ausführen (wie Cron)

Simple Event Correlator

sec.conf Beispiel:

```
type=Pair
```

```
ptype=RegExp
```

```
pattern=NODE (\S+) IF (\S+) DOWN
```

```
desc=Interface $2 is down at node $1
```

```
action=shellcmd notify.sh "%s"
```

```
ptype2=SubStr
```

```
pattern2=node $1 interface $2 up
```

```
desc2=Interface $2 is up at node $1
```

```
action2=shellcmd notify.sh "%s"
```

```
window=86400
```

Links

Informationen

- LogAnalysis.org
- [IETF Syslog Working Group Status Page](#)
- [Syslog-ng FAQ und Loghost HOWTO](#)
- [Artificial Ignorance: How-To Guide](#)

Programme

- [syslog-ng](#)
- [stunnel](#)
- [SEC - simple event correlator](#)
- [Logcheck](#)
- [Swatch](#)
- [Logsurfer](#)