
MailSigger

Karsten Schulz <mailsigger@karstenschulz.name>

Version 0.4

Versionsgeschichte
2007-04-26

KS

Inhaltsverzeichnis

Kurzübersicht	1
Herunterladen	1
Aktuelles	2
Funktionsweise	2
Installation	3
Programmarchiv laden und installieren	3
Benutzer anlegen	4
Arbeitsverzeichnis für MailSigger anlegen:	4
Konfiguration	4
Einstellen der Betriebsparameter	4
Zuordnen der Signaturen zu Absendern	5
Postfix und MailSigger konfigurieren	6
Test der Installation	7
Alternative zu MailSigger	8
Haftungsausschluß und Lizenz	8

Kurzübersicht

MailSigger ist ein kleines Pythonprogramm, welches auf dem ausgehenden Mailserver des Netzwerks installiert werden kann und dort abhängig von bestimmten Absenderadressen Mailsignaturen, auch Haftungsausschlüsse oder "Disclaimer" genannt, an die ausgehenden Mails anhängt. Es kann mit normalen und mit MIME-Mails umgehen und erkennt automatisch die richtige Methode, die Signatur anzuhängen. MailSigger bindet sich gut in einen Postfix-MTA ein, kann aber auch mit anderen MTAs zusammenarbeiten.



Achtung, es wird Python 2.5 vorausgesetzt

MailSigger benötigt ein aktuelles Python 2.5 [<http://www.python.org/>]!

Durch Anpassen der import-Anweisungen kann `mailsigger.py` zwar auch mit älteren Python-Version zum laufen gebracht werden (2.3+). Das ist jedoch nicht ausreichend getestet und wird nicht unterstützt.

Herunterladen

- Aktuelles Programmarchiv mit Dokumentation mailsigger-0.4.tar.bz2 [../downloads/mailsigger-0.4.tar.bz2]
- Dokumentation als PDF mailsigger-0.4.pdf

Aktuelles

- [2007-04-25] - MailSigger Version 0.4 veröffentlicht
 - Fehlerbehebung
 - verbesserte Verarbeitung von MIME-Mails
- [2007-02-21] - MailSigger Version 0.3 veröffentlicht
- [2007-02-14] - Projektstart

Funktionsweise

MailSigger liest aus der Standardeingabe eine Mail komplett ein. Dann wird aus einer Tabelle aufgrund der Zuordnung von Absenderadresse zu einer bestimmten Signaturdatei der Mail eine Signatur angehängt. Es ist sowohl möglich, einem einzelnen Absender eine individuelle Signatur zuzuweisen, als auch einer ganzen Domain eine allgemeine Signatur. Die Tabelle, in der das festgelegt wird, besitzt zwei Spalten:

- Absenderadresse oder Absenderdomain
- Datei, welche die Mail anzuhängende Signatur enthält

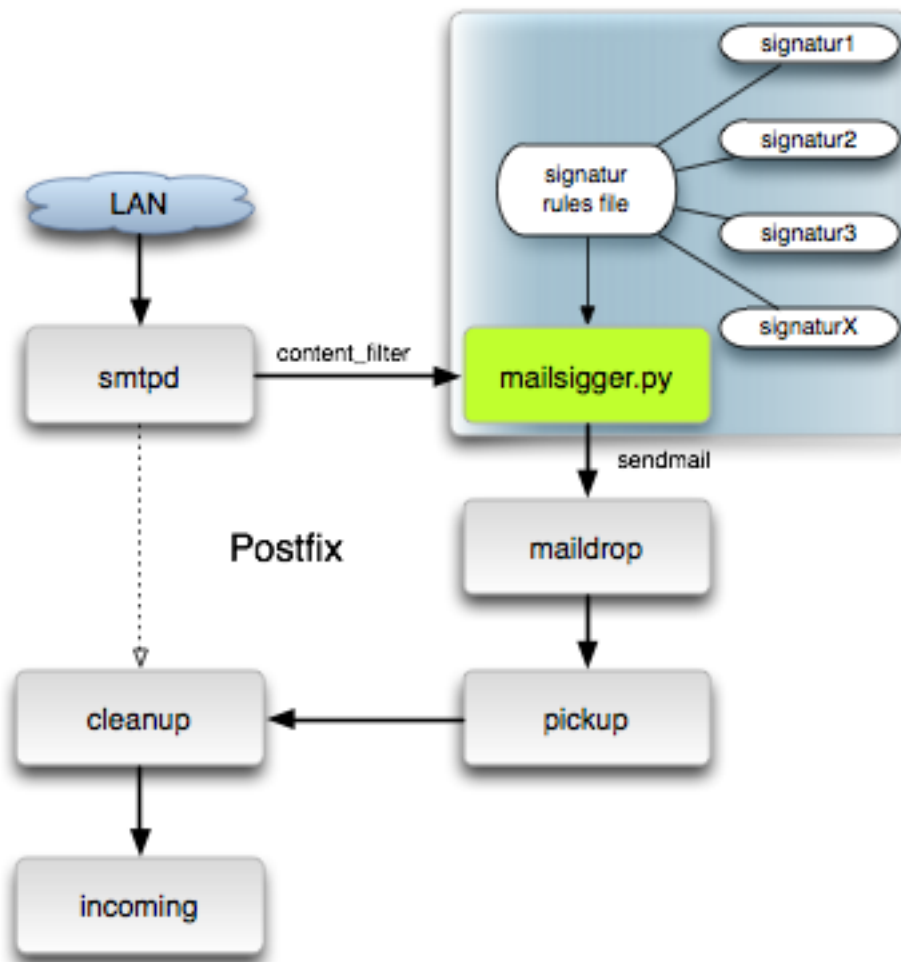
Falls keine Signaturdatei gefunden wird, oder ein Fehler in der Bearbeitung auftrat, wird die Mail nicht verändert. Zum Schluß wird die Mail auf der Standardausgabe ausgegeben und kann so vom MTA weiter transportiert werden.

MailSigger kann sehr einfach in eine vorhandene Postfix Installation integriert werden. Die Zeichnung zeigt, dass Mailsigger durch den Parameter `content_filter` eingebunden werden kann.

Die Filterung durch MailSigger erfolgt unabhängig von einer eventuell bereits vorhandenen Filterung wie beispielsweise durch `spamassassin` oder `amavis`.

MailSigger in der Version 0.4 wurde in 4 kleinen Netzen mit einem gesamten Mailaufkommen von ungefähr 2500 Mails pro Tag getestet.

Abbildung 1. Integration von MailSigger in Postfix



Details zur Einbindung lesen Sie in den folgenden Abschnitten.

Installation

Die Installation geschieht zur Zeit manuell, da das Programmarchiv neben der Dokumentation nur zwei Dateien enthält, die in Ihr System nach `/usr/local/bin/` kopiert werden müssen.

Danach müssen Sie die Signaturregeln und die Signaturdateien angelegen und MailSigger in Ihren MTA integrieren.

Programmarchiv laden und installieren

Nach dem Herunterladen des `mailsigger-0.4.tar.bz2` [`../../downloads/mailsigger-0.4.tar.bz2`] entpacken Sie das Archiv und kopieren die beiden Dateien `mailsigger` und `mailsigger.py` in das Verzeichnis `/usr/local/bin`. Dabei ist die Datei `mailsigger` ein Shellsript, welches aus der Postfix-Dokumentation entnommen wurde und den MTA mit dem Filter verbindet. Das Pythonprogramm `mailsigger.py` ist der eigentliche Filter und wird von dem Shellsript aufgerufen.

```

host:~ # wget http://www.karstenschulz.name/assets/downloads/mailsigger-0.4.tar.bz2
...
host:~ # tar xvjf mailsigger-0.4.tar.bz2
host:~ # cd mailsigger-0.4
host:~ # cp mailsigger* /usr/local/bin
  
```

Benutzer anlegen

Danach legen Sie einen Benutzer an, unter dessen Identität MailSigger arbeiten wird. Dieser Benutzer benötigt keine Shell.

```
host:~ # useradd -s /bin/false mailsigger
```

Nun können wir das Arbeitsverzeichnis anlegen.

Arbeitsverzeichnis für MailSigger anlegen:

MailSigger muss die zu bearbeitenden Mails im Dateisystem zwischenspeichern. Hierzu benötigt er ein Arbeitsverzeichnis mit Schreibrechten. Der Einfachheit halber kann MailSigger in diesem Verzeichniss auch die Protokolldatei ablegen. Damit niemand Unbefugtes Mails lesen kann, setzen wir die Berechtigung des Verzeichnisses sehr restriktiv.

```
host:~ # mkdir /var/spool/mailSigger
host:~ # chown mailsigger /var/spool/mailSigger
host:~ # chmod 0700 /var/spool/mailSigger
```



Ort des Arbeitsverzeichnisses

Falls Sie ein anderes Arbeitsverzeichnis benutzen wollen, müssen Sie dieses in dem Shellsript `/usr/local/bin/mailSigger` in der Variablen `INSPECT_DIR` deklarieren!

Damit sind die alle Komponenten installiert. Nun müssen verschiedene Parameter und der Postfix MTA konfiguriert werden.

Konfiguration

Zum Betrieb benötigen wir eine Signaturregeldatei und die Signaturdateien. Im Script `mailSigger.py` muss angegeben werden, in welcher Datei diese Zuordnungen der Absenderadressen zu den Signaturen steht. Das geschieht in dem Parameter `SIGNATURE_RULES_FILE`.

Außerdem muss angegeben werden, in welcher Kodierung die Signaturdateien vorliegen (Parameter: `SIGNATURE_ENCODING`).

Einstellen der Betriebsparameter

Die einstellbaren Parameter befinden sich ziemlich am Anfang der Datei `/usr/local/bin/mailSigger.py`. Öffnen Sie sie in Ihrem Lieblingseditor:

Beispiel 1. Konfigurationsoptionen in der Datei `/usr/local/bin/mailSigger.py`

```
#####
# please configure the following values: #
#####

# configure different sig-files for different sender here:
SIGNATURE_RULES_FILE = '/etc/postfix/signature_rules'
```

```
# which encoding do you use in your different signature files?
SIGNATURE_ENCODING = 'utf-8'

# loglevel, one of ERROR or DEBUG (atm)
VERBOSITY = logging.DEBUG

# logfile
LOGFILE = '/var/spool/mailSigger/mailSigger.log'

#####
# end of configuration #
#####
```

In diesem Beispiel wird im Parameter `SIGNATURE_RULES_FILE` die Zuordnungstabelle von Absendern und deren Signaturdateien auf die Datei `/etc/postfix/signature_rules` festgelegt.

Da die Signaturen möglicherweise Zeichen enthalten, die nicht im ASCII-Code abgebildet werden können, muss mit dem Parameter `SIGNATURE_ENCODING` die Kodierung des Signatortextes angegeben werden. Auf normalen Systemen ist das `utf-8`. Bei älteren Systemen wird als Kodierung oft auch `iso-8859-1`, und bei einigen Nischensystemen auch `cp1252` benutzt. Schauen Sie in der Dokumentation Ihres Editors nach, in welcher Kodierung dieser Ihre Dateien abspeichert.

Der Parameter `VERBOSITY` legt die Menge der Meldungen fest, die in die Protokolldatei des Programms geschrieben werden soll. Momentan sind folgende Stufen sinnvoll:

Tabelle 1. Protokollstufe

Stufe	Bedeutung
logging.DEBUG	Erzeugt Diagnosemeldungen. Hilfreich zur Fehlersuche.
logging.ERROR	Schreibt nur Fehlermeldungen. Für den Regelbetrieb geeignet

Mit `LOGFILE` legen Sie den Namen der Protokolldatei fest. Zur Zeit wird nur das Protokollieren in eine Datei unterstützt.



Schreibrechte für die Protokolldatei

Achtung, die Benutzer-Id unter der der Filter in Ihrem System laufen wird, muss Schreibrechte auf die angegebene Protokolldatei haben!

Außerdem muss er natürlich Leserechte für die Signaturdateien haben.

Zuordnen der Signaturen zu Absendern

In der Datei `SIGNATURE_RULES_FILE` legen Sie die Zuordnung zwischen einem Absender und die für diesen Absender anzuhängende Signatur fest. In unserem Beispiel wäre das die Datei `/etc/postfix/signature_rules`. Der Aufbau dieser Datei ist denkbar einfach:

Beispiel 2. Zuordnung der Signaturen

```
# Absender          # anzuhängende Signaturdatei
boss@mycompany.com  /etc/postfix/boss-sig.txt
kurt@mycompany.com  /etc/postfix/kurt-sig.txt
no-sig@mycompany.com
@mycompany.com      /etc/postfix/sig-for-all.txt
```

Die Absender `boss@mycompany.com` und `kurt@mycompany.com` bekommen unter ihre ausgehenden Mails jeder seine eigene, individuelle Signatur aus der angegebenen Datei angehängt. Der Absender `no-sig@mycompany.com` bekommt keine Signatur angehängt und durch die letzte Regel bekommen alle anderen Absender aus der Domain `mycompany.com` die allgemeine Signatur aus der Datei `/etc/postfix/sig-for-all.txt` angehängt.

Kommentarzeilen werden mit einer Raute `#` an der ersten Stelle eingeleitet. Leere Zeilen sind ebenfalls erlaubt.

Postfix und MailSigger konfigurieren

Das mitgelieferte Shellsript `mailsigger` ist aus dem hervorragenden `FILTER_README.html` [http://www.postfix.org/FILTER_README.html] der Postfix-Dokumentation entnommen worden. Es dient der Verbindung zwischen Postfix und externen Filterprogrammen, wie `mailsigger.py` eines ist.

Beispiel 3. Konfigurationsoptionen in dem Shellsript `/usr/local/bin/mailsigger`

```
#!/bin/sh

# Simple shell-based filter. It is meant to be invoked as follows:
#   /path/to/script -f sender recipients...

INSPECT_DIR=/var/spool/mailsigger
SENDMAIL="/usr/sbin/sendmail -G -i"
```

Bitte passen Sie die Zeilen mit den Shell-Variablen `INSPECT_DIR` und `SENDMAIL` Ihren Gegebenheiten an. `INSPECT_DIR` ist das oben eingestellte Arbeitsverzeichnis. Falls Sie eine Standardinstallation benutzen, sollten die voreingestellten Werte für Sie funktionieren. Wenn Sie unsicher sind, lesen Sie die Postfix-Dokumentation [http://www.postfix.org/FILTER_README.html].

Nun muss Postfix diese neue Methode der Mailverarbeitung noch bekannt gemacht werden. In der Postfix-Terminologie handelt es sich bei MailSigger um einen neuen Transport. Deshalb konfigurieren wir einen neuen Transport `mailsigger` in der Datei `/etc/postfix/master.cf`:

Beispiel 4. Anpassungen an der `master.cf` von Postfix

```
# =====
# service type  private unpriv  chroot  wakeup  maxproc  command + args
#               (yes)    (yes)    (yes)    (never) (100)
# =====
127.0.0.1:10025 inet      n        -        n        -        -        smtpd
    -o content_filter=
    -o local_recipient_maps=
    -o smtpd_helo_restrictions=
    -o smtpd_client_restrictions=
    -o smtpd_sender_restrictions=
```

```
-o smtpd_recipient_restrictions=permit_mynetworks,reject_unauth_destination
-o mynetworks=127.0.0.0/8

smtp-amavis      unix      -      -      n      -      2      smtp
-o smtp_data_done_timeout=1200
-o disable_dns_lookups=yes

❶ mailsigger    unix      -      n      n      -      -      pipe
  flags=Rq user=mailsigger argv=/usr/local/bin/mailsigger -f ${sender} ${recipient}

❷ 192.168.0.1:smtp      inet  n      -      n      -      -      smtpd
  -o content_filter=mailsigger
127.0.0.1:smtp      inet  n      -      n      -      -      smtpd
```

- ❶ Der neue Transport *mailsigger* ruft das Script `/usr/local/bin/mailsigger` unter der Benutzer-Id *mailsigger* auf
- ❷ Durch diesen Transport werden alle Mails, die über den Socket `192.168.0.1:25` eingeliefert werden, durch MailSigger geschickt. Die IP-Adresse ist natürlich entsprechend anzupassen. Nehmen Sie als IP-Adresse den Socket, auf dem Ihre Clients die Mails zum Versenden einliefern. Alle Mails, die Ihre Clients durch diesen Transport abliefern, laufen so durch MailSigger und danach gegebenenfalls durch den Hauptfilter, den sie möglicherweise in der `main.cf` mit dem Postfix-Parameter `content_filter` festgelegt haben.

Test der Installation

Wenn Sie alle Installationsschritte durchgeführt haben, können Sie einen manuellen Test wie folgt durchführen:

```
host:~ # mailsigger.py < mytestmail
```

Es sollte je nach Konfiguration Ihrer Signaturregeln entweder die unveränderte Testmail auf der Standardausgabe erscheinen, oder eben die Mail mit angehängter Signatur ausgegeben werden.



Eigentümer der Protokolldatei

Wenn dieser erste Test der erste Lauf von `mailsigger.py` war, dann wird die Protokolldatei `/var/spool/mailsigger/mailsigger.log` nun mit größter Wahrscheinlichkeit *root* gehören. Bitte ändern Sie den Eigentümer wieder auf *mailsigger*, damit der Filter Schreibzugriff auf die Protokolldatei hat, wenn er von Postfix aufgerufen wird!

Und testen Sie den Mailversand mit:

```
host:~ # mailsigger -f sender@here receiver@there < mytestmail
```

Wobei das `mailsigger`-Script die Mail verschicken soll, indem es die Mail per `sendmail`-Aufruf wieder per Postfix einliefert. Lesen Sie die Protokolldatei Ihres Postfix aufmerksam durch! Lesen Sie die Protokolldatei von `mailsigger.py` durch. Wenn alles in Ordnung ist, können Sie nun mit einem beherztem

```
host:~ # postfix reload
```

die neue Konfiguration aktivieren. Viel Spaß!

Alternative zu MailSigger

Um Signaturdateien und Haftungsausschlüsse ("disclaimer") an Mails anzuhängen, gibt es auch das Programm alterMIME [<http://www.pldaniels.com/altermime/>] von Paul L. Daniels. Wenn Sie Wert auf höchste Effizienz legen, oder Ihr MTA sehr stark ausgelastet ist, sollte alterMIME Ihre erste Wahl sein, weil es ein in C geschriebenes, schnelles Programm ist.

Haftungsausschluß und Lizenz

Diese Software wurde nach bestem Wissen und Gewissen erstellt, ist mit Sicherheit aber nicht fehlerfrei! Bitte testen Sie Ihre Installation vor Inbetriebnahme! Stellen Sie sicher, dass Sie keine Mails verlieren! Überprüfen Sie regelmäßig die Protokolldatei und das Spool-Verzeichnis! Für Verluste oder Schaden wird keine Haftung übernommen!

Diese Software unterliegt der Lizenz GPL Version 2 [<http://www.gnu.org/licenses/gpl.html>]

Dieses Dokument wurde mit Hilfe von Stuart Rackham's asciidoc [<http://www.methods.co.nz/asciidoc/>] erstellt.

Sparen Sie nicht mit Kritik und Verbesserungsvorschlägen! [<mailto:mailsigger@karstenschulz.name>]