Scripte

- anzahl_connections.pl
- apanonymizer.pl
- check_radius.pl
- check_telnet_response.pl
- ldap-telefonbuch.pl
- scp-on-xferlog.pl
- show-record.pl
- Spam vorlesen lassen mit espeak
- mail_from_perl.pl

anzahl_connections.pl

```
#!/usr/bin/perl
#
# Orginal: http://www.brandonhutchinson.com/Timeout_command.html
#
# 2007 - OV angepasst auf RedHat-Pfade und um die Ausgabe des netstats erweitert
#

use strict;

# Zeige IPs ab $threshold gleichzeitigen Verbindungen
my $threshold = 5;
my %cmd_read;

foreach (`/bin/ps axuww | /bin/egrep 'sendmail: server.*cmd read'`) {
   $cmd_read{$1}++ if
(/sendmail:\sserver.*\[(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\]\scmd\sread$/);
}

# Subroutine to sort hash by ascending value
sub hashValueAscendingNum {
   $cmd_read{$a} <=> $cmd_read{$b};
}

print "Liste der IPs mit mehr als $threshold Verbindungen in Status CMD READ\n\n";

# Print sorted results
foreach my $key (sort hashValueAscendingNum (keys(%cmd_read))) {
   printf "%-15s (%-d)\n", $key, $cmd_read{$key} if
      ($cmd_read{$key} >= $threshold);
}

print "\naktuelle Verbindungen auf Port 25: ";
system("netstat -an | grep \"212.34.175.249:25\" | grep \"ESTABLISHED\" | wc -l");
```

apanonymizer.pl

Dieses Script ersetzt das letzte Byte einer IP-Adresse durch 0 (/24er Maske).

In der httpd.conf ist die Zeile mit dem combined-Logformat durch folgende zu ersetzen:

```
LogFormat "%a %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
```

Nun wird in der Apache-Konfiguration entweder global, bzw. wenn vorhanden fuer jeden vhost der ErrorLog- und CustomLog-Eintrag angepasst:

```
ErrorLog "|/usr/local/bin/apanonymizer.pl /var/log/httpd/error_log"
CustomLog "|/usr/local/bin/apanonymizer.pl /var/log/httpd/access_log" combined
```

Änderungen an Logrotation-Scripts etc. sind nicht erforderlich. Bei vielen Vhosts kann es möglicherweise zu Performanceproblemen kommen, da für jeden Vhost jeweils 2 Instanzen des Scripts gestartet werden. Ich habe das Script ohne Probleme auf einem Server mit knapp 100 Vhosts getestet (allerdings nicht unter Volllast).

Jetzt nur noch das folgende Script wird unter /usr/local/bin/apanonymizer.pl ablegen und den Apache neu starten.

```
#!/usr/bin/perl
#
# Apache-Logfile Anonymisierung
#
# 2008 magenbrot <scripts@magenbrot.net>
#

use strict;
use warnings;
use IO::Handle;

if (@ARGV != 1) { exit 1; }

my $LOG;
open($LOG, ">>", $ARGV[0]) or die("Could not open $ARGV[0]: $!\n");
$LOG->autoflush(1);

while (my $line = <STDIN>) {
  chomp $line;
  if ($line =~ /^\[.*/) {
    # error_log
```

```
    $line =~ s/^(.*?\[client \d+\.\d+\.\d+)\.\d+(\].*)/$1.0]$2/;
  } else {
    # access_log
    $line =~ s/^(\d+\.\d+\.\d+)\.\d+/$1.0/;
    $line =~ s/""$/\"/;
  }
  print($LOG "$line\n");
}
```

check_radius.pl

```
#!/usr/bin/perl

use Authen::Radius;

my $radiusserver = $ARGV[0];
my $radiussecret = $ARGV[1];
my $username = $ARGV[2];
my $password = $ARGV[3];
my $timeout = $ARGV[4];
my $debug = $ARGV[5] ? 1:0;

if (! $ARGV[4]) {
        print "Usage: check_radius.pl <server> <secret> <user> <password> <timeout>
[<verbose>]\n";
        exit 0;
}

my $r = new Authen::Radius(Host => $radiusserver, Secret => $radiussecret, TimeOut =>
$timeout, Debug => $debug);

if (! $r) {
        print "CRITICAL - Radius dead\n";
        exit 2;
}

if ($r->check_pwd($username,$password)) {
        print "OK - Check ok\n";
        exit 0;
} else {
        print "Error: ", $r->strerror(), "\n" if $debug;
        print "Error: ", $r->get_error(), "\n" if $debug;
        print "OK - but Authen-Check failed\n";
        exit 1;
}
```

check_telnet_response.pl

```
#!/usr/bin/perl

#
# checked Telnet-Port unserer Portmaster auf korrekte Response
#
# 2005 by OV
#

use IO::Socket;

my $line = "";

my $RemoteHost = $ARGV[0];
my $RemotePort = $ARGV[1];
my $CheckString = $ARGV[2];

local $SIG{ALRM} = sub { print "CRITICAL - Timeout waiting for correct response\n"; exit 2; };
alarm 6;

# print "Host " . $RemoteHost . " Port " . $RemotePort . " String " . $CheckString . "\n";

my $remote = IO::Socket::INET->new(
        Proto => "tcp",
        PeerAddr => $RemoteHost,
        PeerPort => $RemotePort,
        Timeout => 10,)
or die "CRITICAL - cannot connect to Port $RemotePort at $RemoteHost";

while (($line = <$remote>) && !($line =~ /$CheckString/)) {
        # dummy-schleife solange durchlaufen bis der checkstring auftaucht
}

print "OK - Response received\n";
exit 0;
```

ldap-telefonbuch.pl

```
#!/usr/bin/perl

#
# 2007 Oliver Voelker <info(at)ovtec.it>
#

use strict;
use Net::LDAP;
use Getopt::Long;

my $ldapserver = "ldap.mein-server.de";
my $base       = "ou=People,dc=mein-server,dc=de";

my $debug = 0;
my $search = "*";
my $verbose = 0;
my $help = 0;
my $opts = GetOptions("debug|d!" => \$debug, "verbose|v!" => \$verbose, "search|s:s" =>
\$search, "help|h!" => \$help);

sub LDAPsearch {
  my ($ldap,$searchString,$attrs,$base) = @_;
  if (!$base) { $base = "dc=mein-server,dc=de"; }
  if ($searchString eq "cn=***") { $searchString = "cn=*"; }
  if (!$attrs) { $attrs = [ 'cn','mail' ]; }
  my $result = $ldap->search(base => "$base", scope => "sub", filter => "$searchString", attrs
=> $attrs);
}

my $ldap = Net::LDAP->new($ldapserver) or die "$@";
my $mesg = $ldap->bind(version => 3);

my @Attrs = ( ); # request all available attributes to be returned.

if ($help) {
  print "Usage: tel [OPION]... name\n";
```

```
  print "List ldap-Entries\n\n";
  print "  -h, --help\t\tDisplay this page\n";
  print "  -d, --debug\t\tDebug-Mode - RAW-Display all Entries\n";
  print "  -v, --verbose\t\tExtended display\n";
  print "  -s, --search\t\tSearch for name (surname or lastname or parts of the name is
possible\n";
  exit 0;
}

if ($ARGV[0]) { $search = $ARGV[0]; }
my $result = LDAPsearch($ldap, "cn=*$search*", \@Attrs, $base);
my @entries = $result->entries;
my $entr;

foreach $entr (@entries) {
  next if ($entr->get_value("sn") =~ /ldap$/); # den LDAP-User ausblenden
  if ($debug) {
    print "DN: ", $entr->dn, "\n";
    my $attr;
    foreach $attr (sort $entr->attributes) {
      next if ($attr =~ /;binary$/);
      print "  $attr : ", $entr->get_value($attr) ,"\n";
    }
    print "#------------------------------\n";
  } else {
    my $attr;
    foreach $attr (sort $entr->attributes) {
      next if ($attr =~ /;binary$/);
      if ($attr =~ /cn$/) { print " Name:\t ", $entr->get_value($attr) ,"\n";}
      if ($attr =~ /telephoneNumber$/) { print " Tel:\t ", $entr->get_value( $attr ) ,"\n"; }
      if ($verbose) {
        if ($attr =~ /mail$/) { print " Mail:\t ", $entr->get_value($attr) ,"\n"; }
        if ($attr =~ /facsimileTelephoneNumber$/) { print " Fax:\t ", $entr->get_value($attr)
,"\n"; }

        if ($attr =~ /labeledUri$/) { print " URL:\t ", $entr->get_value($attr) ,"\n"; }
        if ($attr =~ /mobile$/) { print " Mobil:\t ", $entr->get_value($attr) ,"\n"; }
        if ($attr =~ /nsAIMid$/) { print " ICQ:\t ", $entr->get_value($attr) ,"\n"; }
        if ($attr =~ /postalAddress$/) { print " Adr:\t ", $entr->get_value($attr) ,"\n"; }
      }
```

```
    }
    print "#------------------------------\n";
  }
}
```

scp-on-xferlog.pl

Dieses Programm verabschiedet sich direkt nach dem Start in den Hintergrund (Dämon) und schaut dann auf Veränderungen im File /var/log/xferlog. Wird nun via FTP eine Datei hochgeladen, wird sie via SCP (passwortloses Login via Keys sollte vorher natürlich eingerichtet sein) auf die Server in @hosts kopiert. Dies ist z.B. nützlich wenn man mehrere Server in einem Loadbalancingcluster betreibt.

```
#!/usr/bin/perl
#
# 2008 Oliver Voelker <wiki(at)magenbrot.net>
#
# Dieses Script lauscht auf neue Eintraege in /var/log/xferlog und kopiert neu hochgeladene
Dateien
# mit einer Verzoegerung von max. etwa 1-5 Sekunden auf die konfigurierten Hosts.
#

use strict;
use warnings;
use File::Tail;
use File::Basename;
use POSIX qw(setsid);

my $debug = 1;                              # 1 = normal, 2 = extended logging
my $name = "/var/log/xferlog";              # watch this file
#my $logfile = "/dev/null";                 # logfile
my $logfile = "/var/log/scp-on-xferlog";    # logfile
my @hosts = ("host1.de", "host2.de", "host3.de"); # list of hosts

# don't edit anything below this line
my $line = "";

sub daemonize {
  chdir("/") or die("Can't chdir to /: $!");
  open(STDIN, "/dev/null") or die("Can't read /dev/null: $!");
  open(STDOUT, ">>$logfile") or die("Can't write to $logfile: $!");
  open(STDERR, ">>$logfile") or die("Can't write to $logfile: $!");
  defined(my $pid = fork) or die("Can't fork: $!");
  exit if($pid);
  setsid or die("Can't start a new session: $!");
  umask(0);
```

```
}

sub sharefile {
  my ($host,$file) = @_;
  my $try = 0;
  my $error = 1;
  until ($error eq "" || $try == 5) {
    print localtime(time) . " Trans:  $host" if($debug);
    system ("scp -1 \"$file\" root\@$host:$file >/dev/null 2>&1");
    if ($? != 0 && $try eq 0) {
      my $dirname = dirname($file);
      print " - Error:  Creating non-existent directory: $dirname\n" if ($debug >=2);
      system ("ssh -1 root\@$host \"mkdir -p $dirname\" >/dev/null 2>&1");
      system ("scp -1 \"$file\" root\@$host:$file >/dev/null 2>&1");
      $try++;
    } elsif ($? != 0 && $try >= 0) {
      $error=$error . " " . $host;
      print " - failed: $!\n" if ($debug);
      sleep 2;
      $try++;
    }
    else {
      print " - ok\n" if ($debug);
      $error="";
    }
  }
  return $error;
}

# flush the buffer
$| = 1;

# daemonize the program
&daemonize;

my $log = File::Tail->new(name => $name, maxinterval => 5, adjustafter => 10);
while (defined($line = $log->read)) {
  my (undef, undef, undef, undef, undef, undef, undef, undef, $file) = split(/ /, $line);
  my $count = 0;
  my $error = "1";
```

```
  until (-f $file || $count eq 5) {
    # we'll wait max 5*2 seconds for the transfer of the file to complete
    print localtime(time) . "Info:   File does not exist yet: $file\n";
    sleep 2;
    $count++;
  }

  my
($dev,$ino,$mode,$nlink,$uid,$gid,$rdev,$size,$atime,$mtime,$ctime,$blksize,$blocks)=stat($fil
e) if ($debug >= 2);
  print localtime(time) . "File:   $file Size: $size Access: " . scalar localtime($atime) . "
Modified: " . scalar localtime($mtime) . "\n" if ($debug >= 2);

  # copy files
  foreach (@hosts) {
    $error=sharefile($_,$file);
  }

  if ($error eq "") {
    print localtime(time) . " Shared: $file\n" if ($debug);
  } else {
    print localtime(time) . " Error:  $file was not copied to: $error\n";
  }
}
```

nützlich ist dann noch das Initscript, welches sich um den Start des Dämons beim Booten kümmert:

```
#!/bin/bash
#
# scp-on-xferlog        Starts scp-on-xferlog.
#
#
# chkconfig: 2345 12 88
# description: scp-on-xferlog is used to copy new transferred files (from xferlog) to other
configured hosts
#
### BEGIN INIT INFO
# Provides: $scp-on-xferlog
### END INIT INFO
```

```
# Source function library.
. /etc/init.d/functions

[ -f /usr/local/sbin/scp-on-xferlog.pl ] || exit 0

RETVAL=0

start() {
 echo -n $"Starting scp-on-xferlog: "
daemon /usr/local/sbin/scp-on-xferlog.pl
RETVAL=$?
echo
[ $RETVAL -eq 0 ] && touch /var/lock/subsys/scp-on-xferlog
return $RETVAL
}

stop() {
echo -n $"Shutting down scp-on-xferlog: "


killproc scp-on-xferlog.pl
echo
RETVAL=$?
[ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/scp-on-xferlog
return $RETVAL
}

rhstatus() {
status scp-on-xferlog.pl
}

restart() {
stop
start
}

case "$1" in
  start)
  start
```

```
;;
  stop)
  stop
;;
  status)
  rhstatus
;;
  restart)
  restart
;;
  *)
echo $"Usage: $0 {start|stop|status|restart}"
exit 1
esac

exit $?
```

und die Datei für's Logrotate:

```
/var/log/scp-on-xferlog {
    daily
    missingok
    notifempty
    postrotate
/etc/init.d/scp-on-xferlog restart
    endscript
}
```

show-record.pl

```
#!/usr/bin/perl

#
# gibt zu den in "domainlist.txt" angebenen Domains(eine pro Zeile) den zugehoerigen A- und
MX-Record aus
#

use strict;
use Net::DNS;

my $res = Net::DNS::Resolver->new;

open(LISTE,"<domainlist.txt") || die("Konnte die Datei nicht oeffnen!");
while(<LISTE>) {
  chop;
  my $domain = $_;
  my $query = $res->search($domain);
  my @mx = mx($res, $domain);
  print "-----------------------------------------\n";
  print "Domain: $domain\n";
  if ($query) {
    foreach my $rr ($query->answer) {
      next unless $rr->type eq "A";
      print "A:  " . $rr->address, "\n";
    }
  } else {
    warn "query failed: ", $res->errorstring, "\n";
  }

  if (@mx) {
    foreach my $rr (@mx) {
      print "MX: " . $rr->preference, " ", $rr->exchange, "\n";
    }
  } else {
    warn "Can't find MX records for $domain: ", $res->errorstring, "\n";
  }
```

```
}
close(LISTE);
```

und die Domainliste sieht z.B. so aus:

```
heise.de
schnurr.de
bla.de
```

Spam vorlesen lassen mit espeak

Voraussetzungen:

- Perlmodule: Net::IMAP::Simple, Email::Simple
- espeak-Installation

getestet unter Ubuntu 10.10 Maverick

Dieses Script logt sich in eine IMAP-Mailbox ein und liest den Betreff aller Mails im SPAM/Junk-Ordner via espeak vor. Die Parameter sollten natürlich entsprechend angepasst werden.

```
#!/usr/bin/perl

use strict;
use warnings;
use Net::IMAP::Simple;
use Email::Simple;

my $server = "mail.domain.de";
my $user = "username";
my $pass = "passwort";
my $folder = "INBOX/Junk";

my $espeak_params = "-s 160";

# open a connection to the IMAP server
my $imap = Net::IMAP::Simple->new($server) || die "Unable to connect to IMAP:
$Net::IMAP::Simple::errstr\n";

# login
if(!$imap->login($user, $pass)) {
   print STDERR $imap->errstr . "\n";
   exit(64);
}

# select the SPAM-folder
my $nm = $imap->select($folder);

for(my $i = 1; $i <= $nm; $i++){
  my $es = Email::Simple->new(join '', @{ $imap->top($i) } );
```

```
  if($imap->seen($i)){
    # already read mail
    print "*";
  } else {
    # new mail
    print " ";
    # uncomment this to only read new mail
    #my @args = ("espeak", $espeak_param,  "\"" . $es->header('Subject') . "\"");
    #system(@args) || die("system @args failed: $?");
  }
  # print the mailnumber and subject
  printf("[%03d] %s\n", $i, $es->header('Subject'));

  # comment this out, when you've enabled the read of unread mails only above
  my @args = ("espeak", $espeak_params,  "\"" . $es->header('Subject') . "\"");
  system(@args) || die("system @args failed: $?");
}

$imap->quit;
```

mail_from_perl.pl

mit folgendem Script kann schnell und einfach eine Mail verschickt werden:

```
$RCPT="empfaenger\@domain.de";
$EMAIL = "/usr/sbin/sendmail";
open (MAIL,"|$EMAIL $RCPT");
print MAIL ("From: root\@meinserver.de\n");
print MAIL ("Subject: tolle Mail\n\n");
print MAIL ("Dies ist der tolle Inhalt\n");
close (MAIL);
```