RelayedByDialup

How it works

This plugin checks if a mail has been delivered directly from a dialup-IP. It simply uses the last trusted received-line for the IP and the DNS-Names and tries to find out if the IP-Number is somehow coded into the DNS.

Requirements

none.

Installation

Save the two files below in your local configuration directory.

Remarks

- Note that this is my first SA plugin, so any feedback is welcome
- This Plugin can only be a hint about spam, so don't set the score to high.

– Author: Lars Uffmann, lu ~~at~~ cachescrubber ~~dot~~ org, converted to a module by Cord Beermann, cord@Wunder-Nett.org

Code

relayed_by_dialup.cf

```
loadplugin RelayedByDialup /etc/spamassassin/relayed_by_dialup.pm
header RELAYED_BY_DIALUP        eval:relayed_by_dialup()
describe RELAYED_BY_DIALUP      Sent directly from dynamic IP address
score RELAYED_BY_DIALUP         1
```

relayed_by_dialup.pm

```
# written by Lars Uffmann <lu -at- cachescrubber -dot- org>
# converted to a SA-module by Cord Beermann <cord@Wunder-Nett.org>
# Licence: same as Spamassassin

package RelayedByDialup;
use Mail::SpamAssassin;
use Mail::SpamAssassin::Plugin;
use List::Util qw(sum);
our @ISA = qw(Mail::SpamAssassin::Plugin);

$dbg_text = 'dynamic_relay: ';

sub new {
  my ($class, $mailsa) = @_;
  $class = ref($class) || $class;
  my $self = $class->SUPER::new($mailsa);
  bless ($self, $class);
  $self->register_eval_rule ("relayed_by_dialup");
  return $self;
}

sub relayed_by_dialup {
  my ($self, $permsgstatus) = @_;
  my $match = 0;
  # we need the reveived header from _our_ first MX.
  # we can only match this if we have at least 1 untrusted header
  Mail::SpamAssassin::Plugin::dbg("dynamic_relay: starting");
  unless ($permsgstatus->{num_relays_untrusted} > 0) {
    Mail::SpamAssassin::Plugin::dbg("dynamic_relay: num_relays_untrusted undefined");
    return 0;
```

```
  } else {
    my $relay = $permsgstatus->{relays_untrusted}->[0];
    Mail::SpamAssassin::Plugin::dbg ("dynamic_relay: mx=" . $relay->{by});
    if ($relay->{no_reverse_dns} || $relay->{rdns} eq '' ) {
      Mail::SpamAssassin::Plugin::dbg($dbg_text . "cannot perform, no rDNS");
      return 0;
    }

    if (_is_dynamic_ip($relay->{ip}, $relay->{rdns})) {
      Mail::SpamAssassin::Plugin::dbg($dbg_text . "match: " . $relay->{ip} . "=" . $relay->{rdns});
      return 1;
    }
    Mail::SpamAssassin::Plugin::dbg($dbg_text . "tried: " . $relay->{ip} . "=" . $relay->{rdns});
    return 0;
  }
}

sub _is_dynamic_ip {
  my @ip = split(/\./, $_[0]);
  my $name = $_[1];
  return 0 unless $name;
  # convert addresses in hex to dotted decimal notation.
  $name =~ s/\b([a-f0-9]{8})\b/join(".", unpack("C*", pack("H8", $1)))/eg;
  # try shorter suffixes of $IP
  return 1 if _is_rr_dynamic_ip(join(".",@ip), $name);
  shift(@ip);
  return 1 if _is_rr_dynamic_ip(join(".",@ip), $name);
  return 0;
  # will lead to false positives ...
  shift(@ip);
  return 1 if _is_rr_dynamic_ip(join(".",@ip), $name);
  return 0;
}

sub _is_rr_dynamic_ip {
  my $ip = $_[0];
  my $is_ip = $_[1];
  # remove anything but digits > 0
  $is_ip =~ s/[^1-9]//g;
  # normalize the ip
  my $ip_test = $ip;
  # remove anything but digits > 0
  $ip_test =~ s/[0.]//g;
  my $diff = length($is_ip) - length($ip_test);
  return 0 if $diff < 0;
  my $offset = 0;
  my $ip_test_sum = sum(0, split("", $ip_test));
  do {
    my $test = substr($is_ip, $offset++, length($ip_test));
    my $is_ip_sum = sum(0, split("", $test));
    return 1 if $is_ip_sum == $ip_test_sum;
  } while ($diff--);
  return 0;
}

1;
```