URICountryPlugin

URICountry Plugin

This SpamAssassin plugin module provides meta data for scoring URIs based on the country in which they are hosted. This enables rules to be created and scored individually for any valid country code. The module relies on IP::Country::Fast which has the advantage that it does not require any DNS lookups. IP::Country::Fast can be obtained from CPAN.

Warning

This Plugin has been observed by several people to cause **very slow** message processing on some content, causing a 120s+ timeout.

Usage

Add the following to your local.cf file:

```
loadplugin Mail::SpamAssassin::Plugin::URICountry

uricountry      URICOUNTRY_XX   XX
header          URICOUNTRY_XX   eval:check_uricountry('URICOUNTRY_XX')
describe        URICOUNTRY_XX   Contains a URI hosted in XX
tflags          URICOUNTRY_XX   net
score URICOUNTRY_XX 2.0
```

Where XX is replaced with the 2 character country code of your choice. (e.g. CN, KR, RO, RU, IN etc.)

The Code

```
=head1 NAME

URICountry - add message metadata indicating the country code of each relay

=head1 SYNOPSIS

  loadplugin     Mail::SpamAssassin::Plugin::URICountry

=head1 REQUIREMENT

This plugin requires the IP::Country::Fast module from CPAN.

=cut

package Mail::SpamAssassin::Plugin::URICountry;

use Mail::SpamAssassin::Plugin;
use strict;
use bytes;

use vars qw(@ISA);
@ISA = qw(Mail::SpamAssassin::Plugin);

# constructor: register the eval rule
sub new {
  my $class = shift;
  my $mailsaobject = shift;

  # some boilerplate...
  $class = ref($class) || $class;
  my $self = $class->SUPER::new($mailsaobject);
  bless ($self, $class);

  $self->register_eval_rule ("check_uricountry");

  return $self;
}
```

```
# this is just a placeholder; in fact the results are dealt with later
sub check_uricountry {
  my ($self, $permsgstatus, $rulename) = @_;
  return 0;
}

# and the eval rule itself
sub parsed_metadata {
  my ($self, $opts) = @_;
  my $scanner = $opts->{permsgstatus};

  my $reg;

  eval {
    require IP::Country::Fast;
    $reg = IP::Country::Fast->new();
  };
  if ($@) {
    dbg ("failed to load 'IP::Country::Fast', skipping");
    return 1;
  }

  my %domlist = ();
  foreach my $uri ($scanner->get_uri_list()) {
    my $dom = my_uri_to_domain($uri);
    if ($dom) {
        dbg("debug: URICountry $uri in $dom");
        $domlist{$dom} = 1;
    }
  }

  # Build a list of the countries for URIs in the message.
  my %countries = ();
  foreach my $dom (keys(%domlist)) {
    my $cc = $reg->inet_atocc($dom) || "XX";
    dbg("debug: URICountry $dom in $cc");
    $countries{lc($cc)} = 1;
  }

  # Now check if any match any defined rules.
  foreach my $rule (keys(%{$scanner->{conf}->{uricountry}})) {
    my $country = lc($scanner->{conf}->{uricountry}->{$rule});
    if($countries{$country}) {
      dbg ("debug: URICountry hit rule: $country");
      $scanner->got_hit($rule, "");
    }
  }

  return 1;
}

sub parse_config {
  my ($self, $opts) = @_;

  my $key = $opts->{key};

  if ($key eq 'uricountry') {
    if ($opts->{value} =~ /^(\S+)\s+(\S+)\s*$/) {
      my $rulename = $1;
      my $country = $2;

      dbg("debug: URICountry: registering $rulename");
      $opts->{conf}->{uricountry}->{$rulename} = $country;
      $self->inhibit_further_callbacks(); return 1;
    }
  }

  return 0;
}

# Taken from the one in Util.pm but we don't want to drop the hostname doing so
```

```
# often leaves us with no A record.
sub my_uri_to_domain {
  my ($uri) = @_;

  # Javascript is not going to help us, so return.
  return if ($uri =~ /^javascript:/i);

  $uri =~ s,#.*$,,gs;                   # drop fragment
  $uri =~ s#^[a-z]+:/{0,2}##gsi;        # drop the protocol
  $uri =~ s,^[^/]*\@,,gs;               # username/passwd
  $uri =~ s,[/\?\&].*$,,gs;             # path/cgi params
  $uri =~ s,:\d+$,,gs;                  # port

  return if $uri =~ /\%/;         # skip undecoded URIs.
  # we'll see the decoded version as well

  # keep IPs intact
  if ($uri !~ /^\d+\.\d+\.\d+\.\d+$/) {
    # ignore invalid domains
    return unless (Mail::SpamAssassin::Util::RegistrarBoundaries::is_domain_valid($uri));
  }

  # $uri is now the domain only
  return lc $uri;
}

sub dbg { Mail::SpamAssassin::dbg (@_); }

1;
```