ReplaceTags

The ReplaceTags Plugin

This SpamAssassin plugin module allows you to create various character-classes and use them in your rules. The basic advantage is that you don't have to write huge rules, which are harder to debug, but can use more readable shortcuts. Another one is that it gets pretty easy to extend rules by only modifying the class. It is also pretty customizeable for individual start/end tags and which tests are parsed.

NOTE: this plugin is included with SpamAssassin 3.1.0 by default.

Code

Add the following to your local.cf file:

```
   loadplugin            ReplaceTags  /path/to/plugin/ReplaceTags.pm
  replace_rules         VIAGRA_OBFU CIALIS_OBFU # ... and so on
  replace_start         <
  replace_end           >

  # Example classes
  replace_class         A               [\@\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xe4\xe3\xe2\xe0\xe1\xe2\xe3\xe4\xe5\xe6]
  replace_class         G               [gk]
  replace_class         I               [il\|\!1y\?\xcc\xcd\xce\xcf\xec\xed\xee\xef]
  replace_class         R               [r3]
  replace_class         V               (?:[vu]|\\\/)
  replace_class         SP              [\s\d\_\-\*\$\%\(\)\,\.\:\;\?\!\}\{\[\]\|\/\?\^\#\~\xa1\\`\'\+]

  # Small functional example
  body                  VIAGRA_OBFU     /(?!viagra)<V>+<SP>*<I>+<SP>*<A>+<SP>*<G>+<SP>*<R>+<SP>*<A>+/i
  describe              VIAGRA_OBFU     obfuscated match "viagra"
  score                 VIAGRA_OBFU     3

  body                  VIAGRA          /viagra/i
  describe              VIAGRA          match plain "viagra"
  score                 VIAGRA          1.5
```

ReplaceTags.pm

```
=head1 NAME

ReplaceTags - Create character-classes for SpamAssassin-rules

A character-class may contain of any character which is valid in a regular expression.
The grouped characters are easy to maintain and the rules are more readable.

=head1 SYNOPSIS

  loadplugin            ReplaceTags  /path/to/plugin/ReplaceTags.pm
  replace_rules         VIAGRA_OBFU CIALIS_OBFU
  replace_start         <
  replace_end           >

  replace_class         A               [\@\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xe4\xe3\xe2\xe0\xe1\xe2\xe3\xe4\xe5\xe6]
  replace_class         G               [gk]
  replace_class         I               [il\|\!1y\?\xcc\xcd\xce\xcf\xec\xed\xee\xef]
  replace_class         R               [r3]
  replace_class         V               [v\\\/wu]
  replace_class         SP              [\s\d\_\-\*\$\%\(\)\,\.\:\;\?\!\}\{\[\]\|\/\?\^\#\~\xa1\<B4>\`\'\+]

  body                  VIAGRA_OBFU     /(?!viagra)<V>+<SP>*<I>+<SP>*<A>+<SP>*<G>+<SP>*<R>+<SP>*<A>+/i
  describe              VIAGRA_OBFU     obfuscated match "viagra"
  score                 VIAGRA_OBFU     3

  body                  VIAGRA          /viagra/i
  describe              VIAGRA          match plain "viagra"
  score                 VIAGRA          1.5
```

```
This example displays how this plugin can be used to match obfuscated and "normal" phrases, which makes it
easier
to increase the scores for rules matching only obfuscated patterns on a real world example.

=head1 CONFIGURATION

=over 4

=item replace_rules     list_of_tests

Specify a list of symbolic test names (seperated with whitespaces) of tests which should be parsed. The test
will only be parsed if it is a body, heade
r, uri, full or raw test.

=item replace_start sign

=item replace_end   sign

Character(s) which indicate the start and end of a class inside a rule. If the class is not enclosed by this
signs it won't be found nor replaced. If you encounter problems run spamassassin from the commandline with
the -D flag and check the output. The default values are < (for replace_start_sign) and > (for
replace_end_sign).

=item replace_class classname expression

Define a character class or a valid regular expression. Valid characters are the same as in any usual regular
expression. It is not a good idea to put
quantifiers inside the character class, better use them inside your rule is shown above in the example.

=cut

package ReplaceTags;

use strict;
use Mail::SpamAssassin;
use Mail::SpamAssassin::Plugin;
use Switch;

our @ISA = qw|Mail::SpamAssassin::Plugin|;

sub new {
  my ($class, $mailsa) = @_;
      $class = ref($class) || $class;

  my $self = $class->SUPER::new($mailsa);

  bless ($self, $class);

  return $self;
}

sub check_start {
  # This is the point where the rulesets get replaced with our stuff
  my ($self,$pms) = @_;

  my $start_tag   = $self->{replace_start};

  my $end_tag     = $self->{replace_end};

  # Put the names of the rules, which get replaced in a hash, for easier usage
  my %Rules;

  for my $rule_name (@{$self->{rules_to_replace}}) {
    $Rules{$rule_name} = 1;
  }

  for my $rule_set (qw|body_tests rawbody_tests head_tests full_tests uri_tests|) {

    # TODO check what that 0 is for (I guess 0 are the GLOBAL tests and user-tests will
    #      have a number (uid?)...guess is wrong :( )
```

```
    while (my ($rule_name, $rule_content) = each %{$pms->{permsgstatus}->{conf}->{$rule_set}->{0}}) {

      # If rulename doesn't match, skip
      next unless $Rules{$rule_name};

      dbg("ReplaceTags: parse rule $rule_name");

      my %SkippedTags;     # Check-Container to prevent endless loops
                           # If a tag isn't found for max 4 times the while-loops stops
                           # this seems to be a _really_ bad idea... This can also occur
                           # if somebody has content inside his classes which contains the
                           # start- and end-tag unquoted, but without a valid tag-name. In that
                           # case this loop, would run for a while...

      while ($pms->{permsgstatus}->{conf}->{$rule_set}->{0}->{$rule_name} =~ /$start_tag(.+?)$end_tag/) {

        my $tag_name = $1;

        dbg("ReplaceTags: Please quote your start and end-signs in rule $rule_name") and last if $SkippedTags
{$tag_name} > 5;

        dbg("ReplaceTags: TagName = $tag_name");

        # If the tag exists, replace it with the corresponding phrase
        if ($tag_name) {
          dbg("Davor => $pms->{permsgstatus}->{conf}->{$rule_set}->{0}->{$rule_name}");
          $pms->{permsgstatus}->{conf}->{$rule_set}->{0}->{$rule_name} = replace_rule
($self,$tag_name,$rule_content,$rule_name);
          dbg("Danach => $pms->{permsgstatus}->{conf}->{$rule_set}->{0}->{$rule_name}");
        }
        else {
          dbg("ReplaceTags: unknown tag encountered ($start_tag$tag_name$end_tag) in rule $rule_name");
          $SkippedTags{$tag_name}++;
        }

      } # while ($pms->{permsgstatus}->{conf}->{$rule_set}->{0}->{$rule_name} =~ /$start_tag(.+?)$end_tag/)

    } while (my ($rule_name, $rule_content) = each %{$pms->{permsgstatus}->{conf}->{$rule_set}->{0}})

  } # for my $test (qw|body_tests rawbody_tests head_tests full_tests uri_tests|)

} # sub check_start

sub replace_rule {

  my ($self,$tag_name,$rule_content,$rule_name) = @_;
  my $start_tag   = $self->{replace_start};
  my $end_tag     = $self->{replace_end};

  my $replacement = $self->{replace_classes}->{$tag_name};

  if ($replacement) {
    # Do the modifications to the rule and return it
    dbg("ReplaceTags: modifying rule $rule_name, replacing $start_tag$tag_name$end_tag with $replacement");
    $rule_content =~ s/$start_tag$tag_name$end_tag/$replacement/g;

    return $rule_content;

  }
  else {
    dbg("ReplaceTags: No replacement found for rule $rule_name, ($start_tag$tag_name$end_tag)");
  }

  # It should never ever get here
  #dbg("ReplaceTags: Please report this to somebody including the full debug-log.");

}

sub parse_config {
  # Used configuration pragmas are
  #   replace_class
```

```
  #   replace_rules
  #   replace_start_sign
  #   replace_end_sign

  my ($self, $opts) = @_;

  switch ($opts->{key}) {

    case 'replace_class'
        {
          if ($opts->{value} =~ /^(\S+)\s+(.*?)\s*$/) {
            my $tag_name        = $1;
            my $tag_replacement = $2;

            dbg("ReplaceTags: parse_config got $tag_name -> $tag_replacement");

            $self->{replace_classes}->{$tag_name} = $tag_replacement;
          }
        }

    case 'replace_rules'
        {
          # The replace_rules configuration-pragma contains a comma separated
          # list of all rules, which should get parsed by this module

          @{$self->{rules_to_replace}} = split(' ', $opts->{'value'});
        }

    case 'replace_start'
         {
           # The replace_start_sign indicates the start of a replacement-tag. If this
           # setting is omitted a < is used as default.
           if ($opts->{value}) {
             dbg("ReplaceTags: setting start sign to '$opts->{value}'");

             $self->{replace_start} = $opts->{value};
           }
           else {
             dbg("ReplaceTags: no start sign specified. Fall back to default '<'");

             $self->{replace_start} = '<';
           }
         }

    case 'replace_end'
         {
           # The replace_end_sign indicates the end of a replacement-tag. If this
           # setting is omitted a > is used as default.
           if ($opts->{value}) {
             dbg("ReplaceTags: setting end sign to '$opts->{value}'");

             $self->{replace_end} = $opts->{value};
           }
           else {
             dbg("ReplaceTags: no end sign specified. Fall back to default '>'");

             $self->{replace_end} = '>';
           }
         }
  }
}

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

1;
```

How To Use It

See the pod or the example above.

Example Classes

```
replace_class   A       [gra\@\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xe4\xe3\xe2\xe0\xe1\xe2\xe3\xe4\xe5\xe60o]
replace_class   B       [b8]
replace_class   C       [kc\xc7\xe7@]
replace_class   D       [d\xd0]
replace_class   E       [e3\xc8\xc9\xca\xcb\xe8\xe9\xea\xeb\xa4]
replace_class   F       [f]
replace_class   G       [gk]
replace_class   H       [h]
replace_class   I       [il\|\!1y\?\xcc\xcd\xce\xcf\xec\xed\xee\xef]
replace_class   J       [j]
replace_class   K       [k]
replace_class   L       [il\|\!1\xa3]
replace_class   M       [m]
replace_class   N       [n\xd1\xf1]
replace_class   O       [go0\xd2\xd3\xd4\xd5\xd6\xd8\xf0\xf2\xf3\xf4\xf5\xf6\xf8]
replace_class   P       [p\xfek]
replace_class   Q       [q]
replace_class   R       [r]
replace_class   S       [s\xa6\xa7]
replace_class   T       [t]
replace_class   U       [uv\xb5\xd9\xda\xdb\xdc\xfc\xfb\xfa\xf9\xfd]
replace_class   V       [v\\\/]
replace_class   W       [wv]
replace_class   X       [x\xd7]
replace_class   Y       [y\xff\xfd\xa5j]
replace_class   Z       [zs]
replace_class   PIC     (jpe*g|gif|png)
replace_class   SP      [\s\d\_\-\*\$\%\(\)\,\.\:\;\?\!\}\{\[\]\|\/\?\^\#\~\xa1\<B4>\`\'\+]
replace_class   CUR     [\$\xa5\xa3\xa4\xa2]
```