

Extending Apache SpamAssassin Using Plugins



Michael Parker

ApacheCon December 2005

<http://people.apache.org/~parker/presentations/>

If you are reading these slides without the notes go to the website now and download them and read them while you follow along with the slides, trust me it will be much better.



SpamAssassin

Introduction

- The Way it Was
- The Way it is Now
- The Basics
- Beyond the Basics
- Examples
- Future Directions



The Way It Was

- Conf.pm
- EvalTests.pm
- SpamAssassin.pm
- Other Ugliness



The Way it is Now

- Create your own module
- Load the module in SpamAssassin
- Profit



Advantages of Plugins

- Cleaner Code
- Ability to Publish New Rules Outside Normal Development Cycle
- Cleaner Code
- Lower Memory When Not In Use
- Cleaner Code
- Lower Barrier to Entry
- Did I Mention Cleaner Code



The Basics

- Creating a Basic Plugin Module
- Loading Your Plugin Module
- Configuration
- Creating and Registering Eval Rules



Creating a Basic Plugin Module

```
package TestPlugin;

use Mail::SpamAssassin::Plugin;

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

sub new {
    my $class = shift;
    my $mailsaobject = shift;

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

    return $self;
}
```



Loading Your Plugin Module

- Plugin modules are loaded via one of these loadplugin configuration options

```
loadplugin Mail::SpamAssassin::Plugin::TestPlugin  
loadplugin TestPlugin  
loadplugin TestPlugin /path/to/TestPlugin.pm
```

- You can place the loadplugin option in any configuration file, however, most plugins are loaded in a *.pre file
- *.pre files are loaded before all other configuration (*.cf) files



Configuration

```
# configuration for our test plugin
ifplugin TestPlugin
    fire_test_eval 1
    header TEST_PLUGIN_RULE eval:test_eval_rule()
endif
```

- Generally plugin configuration is placed in it's own config file, for instance 25_testplugin.cf



Parsing Configuration Options

```
sub parse_config {  
    my ($self, $opts) = @_;  
  
    my $key = $opts->{key};  
  
    if ($key eq 'fire_test_eval') {  
        $opts->{conf}->{fire_test_eval} = $opts->{value};  
        $self->inhibit_further_callbacks();  
        return 1;  
    }  
  
    return 0;  
}
```



Creating an Eval Rule

```
sub test_eval_rule {  
    my ($self, $pms) = @_;  
  
    if ($pms->{conf}->{fire_test_eval} == 1) {  
        return 1;  
    }  
  
    return 0;  
}
```

Eval rules return 1 (true) for a hit or 0 (false) for a non-hit.



Registering an Eval Rule

```
sub new {  
    my $class = shift;  
    my $mailsaobject = shift;  
  
    $class = ref($class) || $class;  
    my $self = $class->SUPER::new($mailsaobject);  
    bless ($self, $class);  
  
    $self->register_eval_rule("test_eval_rule");  
  
    return $self;  
}
```



Beyond The Basics



SpamAssassin

Example Plugin Hooks

`perldoc Mail::SpamAssassin::Plugin`

- `signal_user_changed`
Signals that the current user has switched to a new one
- `check_start`
Signals that a message check operation is starting
- `check_end`
Signals that a message check operation is finished



More Plugin Hooks

- `extract_metadata`
Signals that a message is being mined for metadata
- `parsed_metadata`
Signals that a message's metadata has been parsed and can now be accessed
- `services_authorized_for_username`
Validates that a given username is authorized to use certain services



Parsing Configuration Options

SpamAssassin 3.1

```
sub set_config {  
    my($self, $conf) = @_;  
    my @cmds = ();  
  
    push(@cmds, {  
        setting => 'fire_test_eval',  
        default => 1,  
        type => $Mail::SpamAssassin::Conf::CONF_TYPE_BOOL,  
    });  
  
    $conf->{parser}->register_commands(\@cmds);  
}
```



Parsing Configuration Options SpamAssassin 3.1 Continued

```
sub new {  
    my $class = shift;  
    my $mailsaobject = shift;  
  
    $class = ref($class) || $class;  
    my $self = $class->SUPER::new($mailsaobject);  
    bless ($self, $class);  
  
    $self->register_eval_rule("test_eval_rule");  
  
    $self->set_config($mailsaobject->{conf});  
  
    return $self;  
}
```



Examples

- CustomDeleteTags
- AuthzUser
- ~~Replace Word in Subject~~
- Persistent Database Connections

You can find the full source and documentation for the examples on the SpamAssassin wiki: <http://wiki.apache.org/spamassassin/CustomPlugins> or <http://www.apache.org/~parker/presentations/index.html>



CustomDeleteTag Plugin

=head1 NAME

```
package CustomDeleteTag
```

=head1 SYNOPSIS

```
loadplugin CustomDeleteTag /path/to/CustomDeleteTag.pm
```

```
custom_delete_score 9.5
```

```
add_header all Delete _CUSTOMDELETESCORE_
```

=head1 DESCRIPTION

This SpamAssassin plugin module allows users to specify a value that will be added to the message header, for all messages, specifying what value/score it is safe to delete the message. Obviously, you need some other process that looks at this header and performs the action, since SpamAssassin only filters and does not delete.

=cut



SpamAssassin

```

sub set_config {
    my ($self, $conf) = @_;

    my @cmds = ();
    push(@cmds, {
        setting => 'custom_delete_score',
        default => 1000,
        type => $Mail::SpamAssassin::Conf::CONF_TYPE_NUMERIC,
    });

    $conf->{parser}->register_commands(\@cmds);
}

```

```

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

    $opts->{permsgstatus}->{tag_data}->{CUSTOMDELETESCORE} =
        $opts->{permsgstatus}->{conf}->{custom_delete_score};
}

```



AuthzUser Plugin

=head | NAME

```
package AuthzUser
```

=head | SYNOPSIS

```
loadplugin Mail::SpamAssassin::Plugin::AuthzUser /path/to/AuthzUser.pm  
authzuser_group_file /path/to/htgroup
```

=head | DESCRIPTION

This SpamAssassin plugin module allows you to use a standard HTGroup file to control access to certain services via the `services_allowed_for_username` plugin hook.

The groupfile for this feature looks something like:

```
bayessql: user1 user2 user3
```

=cut



SpamAssassin

```
sub services_allowed_for_username {
    my ($self, $options) = @_;

    my $username = $options->{username};
    my $services = $options->{services};
    my $conf = $options->{conf};
    my $htgroup = Apache::Htgroup->load($conf->{authzuser_group_file});

    foreach my $servicename (keys %{$services}) {
        if ($htgroup->ismember($username, $servicename)) { $services->{$servicename} = 1; }
    }
    return;
}
```



Persistent Database Connections Plugin

- Not every plugin has to do something with a message.
- This shows the power of plugins, do almost anything.
- Tests showed a 15% speed up with PostgreSQL database calls using this plugin.

You can find the full source and documentation for the examples on the SpamAssassin wiki: <http://wiki.apache.org/spamassassin/CustomPlugins> or <http://www.apache.org/~parker/presentations/index.html>



SpamAssassin

Future Directions

- Additional Hooks
- Additional Plugins
- Pluginize Current Functionality
- Alternate Plugins



Where to Get More Info

- Wiki

- <http://wiki.apache.org/spamassassin>

- Mailing Lists

- <http://wiki.apache.org/spamassassin/MailingLists>

- IRC

- #spamassassin on irc.freenode.net

- Updated Slides

- <http://www.apache.org/~parker/presentations/index.html>

