SpamAssassin のプラグイン紹介

日本 SpamAssassin ユーザ会 滝澤 隆史

はじめに

SpamAssassin は様々なスパム判定の手法を用いて総合的にスパムメールの検出を行うメールフィルターです。プラグインを使うことでスパム判定の新しいルールや機能を追加することができます。しかし、実際にプラグインを使おうと思って設定を行ってみると、使い方がよくわからないものがあります。基本的に説明文書が不足していることが多いので、標準配布のルールの記述を見たり、ソースコードを見たりしないと使い方や動作がわからないものがあります。そのため、この文書ではプラグインの紹介と使い方の説明を行います。

また、設定ファイルの記述方法に関する日本語の文書も少ないため、最初に設定ファイルの記述方法を紹介します。筆者の経験に基づいた設定事例も取り混ぜています。プラグインの紹介をすぐに読みたい方は 4 ページから読んでください。

なお、この文書は SpamAssassin 3.1.7 に基づいて記述しています。

設定ファイルの記述方法

設定ファイル

設定ファイルの種類

標準では次の表のような設定ファイルが読み込まれます。

	設定ファイル	説明
1	/etc/mail/spamassassin/*.pre	サイトのプラグイン制御ファイル
2	/usr/share/spamassassin/*.pre /var/lib/spamassassin/VERSION/*.pre	デフォルトのプラグイン制御ファイル
3	/usr/share/spamassassin/*.cf /var/lib/spamassassin/VERSION/*.cf	デフォルトのルールファイル
4	/etc/mail/spamassassin/*.cf	サイトのルールファイル
5	$HOME/.spamassassin/user_prefs	ユーザのルールファイル

1 と 4 がサイトの管理者が設定できるファイルです。local.cf や init.pre などのファイルがあります。サイト共通の設定やプラグインのロードの制御を記述します。

2 と 3 がデフォルトの設定ファイルです。このファイルを編集してはいけません。それぞれ 2 つずつファイルが書いてありますが、上段は sa-update の実行前のもので、下段は sa-update の実行後のものです。ここで VERSION には 3.001007 のようなバージョン番号が入ります。

5 はユーザ毎に個別に設定できるファイルです。

また、設定ファイル中に include オプションを使うことで別の設定ファイルを読み込ませることもできます。

読み込む順番

ファイルを読み込む順番は表の数字の順番通りです。同じオプションやルールがある場合は後から読み込まれたものが上書きします。

include オプションを使った場合は、全てのファイルが読み込まれた後に指定されたファイルが指定元の設定ファイルの順番で読み込まれます。

ここで注意して欲しいのは sa-update 実行後のファイルの読み込みの順番です。sa-update を実行すると/var/lib/spamassassin/VERSION ディレクトリにディレクトリ updates_spamassassin_org が作成され、この中に最新のデフォルトのルールファイルが配置されます。さらに updates_spamassassin_org.pre と updates_spamassassin_org.cf が配置されます。このファイルには include オプションで先のディレクトリ内のルールファイルを読み込む設定が記述されています。そのため、デフォルトのルールファイルがサイトやユーザのルールファイルよりも読み込む順番が後になり、デフォルトの設定を変えようとしてサイトのルールファイルで再定義してもデフォルトの設定に戻されてしまいます。

筆者のお薦めの配置方法

次の表のように site ディレクトリを作成し、その中にルールファイルを用途毎やプラグイン毎に作成します。

設定ファイル	説明
/etc/mail/spamassassin/site.cf	include オプションのみのファイル
/etc/mail/spamassassin/site/*.cf	site.cf により読み込まれるファイル

さらに site.cf ファイルを作成し、次のように include オプションでそれらのファイルを指定します。

```
include site/head_tests.cf
include site/scores.cf
include site/relaycountry.cf
```

local.cf には設定オプションのみを記述します。

このようにすることにより、読み込みの順番の問題も解決しますし、サイトのルールも整理しやすくなります。

必要最小限の設定

SpamAssassin には多くの設定オプションがあります。しかし、以下に述べる必要最小限の設定をサイトのルールファイル local.cf に記述すれば問題なく使えます。この後はお好みの設定を追加してください。

スコアオプション

スパムと判断するために必要なスコアを"required_score"に設定します。デフォルトの設定値は 5 になっていますが、運用当初は次の設定のように大きめに設定した方がよいでしょう。

```
required_score 10.0
```

タグ付けオプション

デフォルトではスパムと判断されたメッセージはレポートメッセージに加工されてしまいます。これを防ぐために次の設定を行います。

```
report_safe 0
```

日本語対応パッチのオプション

日本語対応パッチを導入している場合は次の設定を行います。

```
normalize_charset 1
```

ベイズ学習オプション

ベイズのデータベースはデフォルトではユーザ毎に作られますが、MTA と連携する場合やサイト共通のベイズのデータベースを利用したい場合は次のような設定を行います。

```
bayes_path  /var/spool/spamassassin/bayes
bayes_file_mode 0666
```

テストルールの書き方

パターンテスト

パターンテストに次の表のものがあります。なお、nbody は日本語対応パッチによって追加されたテストです。

対象	説明
header	ヘッダ （MIME 復号化済み）
body	ボディのテキストパートのみ （MIME 復号化済み、HTML タグ等の除去あり）
nbody	ボディのテキストパートのみ（MIME 復号化済み、 HTML タグ等の除去あり、UTF-8 に変換済み）
uri	ボディに記述された URI
rawbody	ボディのテキストパートのみ（MIME 復号化済み）
full	生メッセージ全体（MIME 復号化なし）

header 以外のテストについては次のような形式のルールを記述します。

```
body　　テスト名　　/パターン/修飾子
```

パターンは Perl 正規表現を使うことができます。ボディに"spam"という単語を含むかどうかをテストしたい場合は次のように記述します。

```
body　　SPAM　　/¥bspam¥b/i
```

header テストについては次のような形式のルールを記述します。

```
header　テスト名　　ヘッダ名 op /パターン/修飾子
```

op はマッチすることを意味する"=~"かマッチしないことを意味する"!~"です。Subject ヘッダが"spam"という単語を含むかどうかをテストしたい場合は次のように記述します。

```
header　SUBJECT_SPAM　　Subject =~ /¥bspam¥b/i
```

さらに、ヘッダ名に":"で始まる次の表のようなクエリーを付けることによりパターンテストの対象を変えることができます。なお、:utf8 クエリーは日本語対応パッチの機能です。

クエリー	説明
クエリーなし	MIME 復号化済み
:raw	MIME 復号化なし（生の状態）
:utf8	MIME 復号化済み、UTF8 変換あり
:addr	メールアドレス
:name	名前

例えば、Subject ヘッダが Q エンコードされているかどうかをテストしたい場合は次のように記述します。

```
header　SUBJ_Q_ENCODE　　Subject:raw =~ /=¥?¥S+¥?Q¥?/i
```

header テストにはさらに次のような形式のテストもあります。

```
header　テスト名　　exists:ヘッダ名
```

これは、ヘッダ名のヘッダが存在するかを調べるテストです。例えば、Subject ヘッダがあるかどうかをテストしたい場合は次のように記述します。

```
header　HAS_SUBJECT　　exists:Subject
```

meta テスト

meta テストでは複数のテスト結果を組み合わせて評価することができます。次のような形式のルールを記述します。

```
meta　　テスト名　　ブール演算式
```

ブール演算式には論理積"&&"、論理和"||"、否定"!"、括弧()を使うことができます。

例えば、メーリングリスト用のヘッダがあるかどうかをテストする場合は次のように記述します。

```
header __LIST_ID        List-Id =~ /¥./
header __X_ML_NAME      exists:X-ML-Name
header __MAILING_LIST   exists:Mailing-List
meta MAILING_LIST    __LIST_ID || __X_ML_NAME || __MAILING_LIST
```

なお、"__"（アンダースコア 2 個）で始まるテスト名は単体ではスコアに加算しません。meta テストのサブテストとして用いられます。

パターンテストの注意事項

SpamAssassin はパターンテストのパターンを実行時にほぼそのまま評価しています。そのため、Perl において処理時間がかかるパターンはパターンテストにおいても同様に処理時間がかかります。そのため、筆者がルールを記述するときに気をつけている注意点をいくつか紹介します。

- full テストを使わない。
- body テストにおいて /.*/のような行末までマッチするようなパターンを使わない。
- グループ化するときは後方参照しないように (?:...) 構文を使う。
- (A|B) のような 1 文字の選択であれば [AB] のように集合にする。ただし、日本語の場合はこの方法は使えない。

プラグインの設定

プラグインの制御

プラグインをロードするためには loadplugin オプションを使い、次の書式で記述します。標準のプラグインでない場合は 2 行目のようにパスを指定します。

```
loadplugin プラグインモジュール名
loadplugin プラグインモジュール名　/パス/モジュール.pm
```

SpamAssassin に同梱されているプラグインの制御ファイルは標準では/etc/mail/spamassassin にインストールされます。制御ファイルのファイル名はプラグインが導入されたバージョン毎に異なり、次のようになっています。

init.pre	3.1.0 より前に導入されたプラグイン
v310.pre	3.1.0 で導入されたプラグイン
v312.pre	3.1.2 で導入されたプラグイン

それぞれのファイルで次の例のようにプラグイン毎に loadplugin オプションが記述されています。

```
loadplugin Mail::SpamAssassin::Plugin::TextCat
```

プラグインによってはコメントアウトされていて無効になっているものもあります。有効にする場合は行頭の#を削除してください。

ルールの記述

プラグイン用の設定オプションやルールを記述する場合は次のように ifplugin オプションを使ってください。このようにするとプラグインの使用を一時的に止めたときにも問題が生じません。

```
ifplugin プラグインモジュール名
# この行の間に設定オプションやルールを記述
endif
```

例えば、 AutoLearnThreshold プラグイン用のルールを記述する場合は次のように記述します。

```
ifplugin Mail::SpamAssassin::Plugin::AutoLearnThreshold
bayes_auto_learn_threshold_nonspam      1.0
bayes_auto_learn_threshold_spam         10.0
endif # Mail::SpamAssassin::Plugin::AutoLearnThreshold
```

スコア

スコア

SpamAssassin のデフォルトのルールのスコアの設定の 50%は 1 点以下で、90%は 3 点以下です。ブラックリスト等の特殊なものを除いた場合でも最大のスコアは 4.5 点です。これは小さなスパムらしさの積み上げでスパムであるかどうかを判断しているという SpamAssassin の特徴を表しています。そのため、スコアを設定するときにはデフォルトのルールのスコアとのバランスを考えてください。確実にスパムであると判定できるルールではない場合はスコアを 3 点以下にするのがよいでしょう。

日本語を扱う場合の修正

日本語を扱う場合には、デフォルトのスコアの設定では少し問題が生じます。そのため、以下のようなスコアの再定義を行う必要があります。

```
## 20_body_tests.cfの再定義
score SUBJECT_EXCESS_BASE64 0
score WEIRD_QUOTING 0

## 20_head_tests.cfの再定義
score FROM_EXCESS_BASE64 0
score GAPPY_SUBJECT 0
score SUBJECT_ENCODED_TWICE 0
score SUBJ_ILLEGAL_CHARS 0

## 20_html_tests.cfの再定義
score HTML_COMMENT_8BITS 0
score OBFUSCATING_COMMENT 0

## 20_meta_tests.cfの再定義
score UPPERCASE_25_50 0
score UPPERCASE_50_75 0

## 20_phrases.cfの再定義
score OBSCURED_EMAIL 0
```

なお、このときには「設定ファイル」の項目で記述した設定ファイルの読み込みの順番に注意してください。

自動学習関連

AutoLearnThreshold

説明

このプラグインはベイズの自動学習の可否を閾値により判断する機能を提供します。

"bayes_auto_learn_threshold_nonspam"で設定されたスコアより低いときに、ham（spam ではない）として学習します。デフォルトの設定値は 0.1 です。

"bayes_auto_learn_threshold_spam"で設定されたスコアより高いときに spam として学習します。デフォルトの設定値は 12.0 です。ただし、spam として学習するためにはヘッダテストで 3 点とボディテストで 3 点が最低限必要です。

なお、tflags において"learn", "userconf", "noautolearn"が設定されているルールに関してはスコアの計算は無視されます。

基本設定

インストール時に有効になっているはずですが、v310.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::AutoLearnThreshold
```

ルールの記述例

設定値を変更するためには次のように記述します。

```
ifplugin Mail::SpamAssassin::Plugin::AutoLearnThreshold

bayes_auto_learn_threshold_nonspam      1.0
bayes_auto_learn_threshold_spam         10.0

endif # Mail::SpamAssassin::Plugin::AutoLearnThreshold
```

AWL

説明

SpamAssassin には auto-whitelist（自動ホワイトリスト）という機能があります。このプラグインはこの auto-whitelist の機能を補助し、auto-whitelist を利用できるようにします。この auto-whitelist により送信者毎に過去のスコアを平均化して、メッセージ毎のスコアのばらつきを減らすための補正スコアを算出し、スコアに加算します。

なお、送信者は From ヘッダのメールアドレスと送信元 IP アドレスの組み合わせで識別されます。

基本設定

インストール時に有効になっているはずですが、v310.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::AWL
```

ルールの記述例

基本的にデフォルトのルールのままで使えます。

サイトで共通の auto-whitelist を使う場合は次のように記述します。なお、ファイルを置くディレクトリは予め作っておきます。

```
ifplugin Mail::SpamAssassin::Plugin::AWL

auto_whitelist_path     /var/spool/spamassassin/auto-whitelist
auto_whitelist_file_mode 0666

endif # Mail::SpamAssassin::Plugin::AWL
```

パターンテスト関連

WhitelistSubject

説明

このプラグインは Subject ヘッダのホワイトリストとブラックリストを評価します。

設定オプション"whitelist_subject"と"blacklist_subject"に評価する文字列を設定します。文字列にはファイルグロブのように"*"と"?"が利用できます。

基本設定

インストール時に有効になっているはずですが、v310.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::WhiteListSubject
```

ルールの記述例

基本設定がデフォルトのルールで行われているので、ホワイトリストとブラックリストの登録を行います。

```
ifplugin Mail::SpamAssassin::Plugin::WhiteListSubject

whitelist_subject [Bug *]
blacklist_subject Make Money Fast
blacklist_subject special offer

endif # Mail::SpamAssassin::Plugin::WhiteListSubject
```

MIMEHeader

説明

このプラグインはマルチパートメッセージの各パートの MIME ヘッダに対してパターンテストを行います。

各パートの"Content-Type"や"Content-Transfer-Encoding"や"Content-Disposition"などを評価するのに便利です。もちろん、添付ファイルのファイル名の評価も行えます。

基本設定

インストール時に有効になっているはずですが、v310.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::MIMEHeader
```

ルールの記述例

```
ifplugin Mail::SpamAssassin::Plugin::MIMEHeader

# 画像ファイルが添付されている場合
mimeheader CT_IMAGE Content-Type =~ /image¥//i
describe CT_IMAGE   Content-Type is image/*
score CT_IMAGE      0.5

endif # Mail::SpamAssassin::Plugin::MIMEHeader
```

ReplaceTags

説明

このプラグインはパターンテストのルールにおいて、正規表現のパターンを置き換える機能を提供します。

"spam"という単語を検出したい場合は"/spam/i"のようなパターンを記述しますが、"sp@m"のように単語の文字列中の文字を字形が似たような文字で置き換えられてしまうと、パターンにマッチしなくなります。最近はこのような回避手法がよく用いられており、これに対応するには"/sp[a\@]m/i"のようなパターンを記述する必要があります。しかし、このようなパターンをたくさん記述するとルールの管理が難しくなります。そのため、文字の置き換えを一括して行う機能を提供したのがこのプラグインになります。このプラグインを使うと、先ほどのパターンは"/sp<A>m/i"となり、見通しがよくなります。

ルールファイル 25_replace.cf にデフォルトの設定が記述されているので一度ご覧ください。

基本設定

インストール時に有効になっているはずですが、v310.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::ReplaceTags
```

ルールの記述例

デフォルトの設定を再定義する場合は設定ファイルの読み込みの順番に気をつけてください。

```
ifplugin Mail::SpamAssassin::Plugin::ReplaceTags

### デフォルトのルール 25_replace.cf の再定義
#   デフォルトのreplace_tagの設定は処理が重いので軽くする。
#      s/¥¥x..//g
replace_tag A   [gra¥@&Oo^]
replace_tag B   [b8]
replace_tag C   [ck@]
replace_tag D   d
replace_tag E   [e3]
replace_tag G   [gk]
replace_tag I   [il|!1¥/¥¥fktyj?"]
replace_tag L   [il|!1]
replace_tag M   (?:m|rn)
replace_tag N   n
replace_tag O   [go0]
replace_tag P   p
replace_tag S   [sz]
replace_tag U   [uv]
replace_tag V   (?:[vu]|¥¥¥/)
replace_tag W   [wv]
replace_tag X   (?:x|><)
replace_tag Y   [yj]
replace_tag Z   [zs]
replace_tag SP  [¥s¥d_*¥$¥%(),.:;?!}{¥[¥]|¥/?^¥#~``'+-]

### 任意の0～1文字
replace_inter   A1 .?

### Subjectヘッダにおいてwatchをごまかした単語の検査
header SUBJECT_FUZZY_WATCH      Subject =~ /<inter A1><post
P3>(?!watch)<W><A><T><C><H>/i
describe SUBJECT_FUZZY_WATCH    Attempt to obfuscate words
score SUBJECT_FUZZY_WATCH       2.0
replace_rules SUBJECT_FUZZY_WATCH

### ボディにおいてwatchをごまかした単語の検査
body FUZZY_WATCH        /<inter W1><post
P2>(?!watch)<W><A><T><C><H>/i
describe FUZZY_WATCH    Attempt to obfuscate words
score FUZZY_WATCH       0.5
replace_rules FUZZY_WATCH

endif   # Mail::SpamAssassin::Plugin::ReplaceTags
```

国、言語関連

RelayCountry

説明

このプラグインはメールが経由してきたホストの国コードを調べます。調べた結果はメタデータ"X-Relay-Countries"とタグ"_RELAYCOUNTRY_"に格納されます。この値は header ルールでテストできます。

国コードは二文字で大文字のものが使われます。例えば、"JP"や"KR"です。複数のホストを経由した場合は"JP JP CN"のようにスペース区切りでつなげて格納されます。国コードが見つからなかった場合は"XX"が、プライベートアドレスの場合は"**"が格納されます。

国コードの例をいくつか紹介します。

JP	日本
CN	中国
US	アメリカ合衆国
KR	韓国
MM	ミャンマー
TH	タイ
PL	ポーランド
**	プライベート IP アドレス
XX	国コードが見つからない場合

国コードの一覧は次のサイトで調べてください。

http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/index.html

必要モジュール

IP::Country::Fast

基本設定

init.pre において次の行を有効にしてください。

```
loadplugin Mail::SpamAssassin::Plugin::RelayCountry
```

ルールの記述例

デフォルトのルールはないので、設定を行う必要があります。

```
ifplugin Mail::SpamAssassin::Plugin::RelayCountry

### 特定の国を経由したメッセージにスコアを付ける場合
# この例では中国(CN)のホストを経由する場合
header RELAYCOUNTRY_CN    X-Relay-Countries =~ /CN/
describe RELAYCOUNTRY_CN Relayed via China
score RELAYCOUNTRY_CN     1.5

### 信頼できる国を経由していない場合
# まず、信頼できる国を経由しているかを調べる
# この例では日本(JP)のホストを経由しているかどうかを調べる
header __RELAYCOUNTRY_JP X-Relay-Countries =~ /JP/
# 信頼できる国を経由していない場合を否定演算子を用いて記述する
meta RELAYCOUNTRY_UNTRUSTED     !__RELAYCOUNTRY_JP
describe RELAYCOUNTRY_UNTRUSTED Relayed via untrusted country
score RELAYCOUNTRY_UNTRUSTED    1.0

### メッセージにヘッダを追加する場合
#    追加するヘッダの例) X-Spam-Relay-Countries: CN
add_header all Relay-Countries _RELAYCOUNTRY_

endif   # Mail::SpamAssassin::Plugin::RelayCountry
```

TextCat

説明

このプラグインはメッセージ中のテキストから言語を推定します。推定結果はメタデータ"X-Language"とタグ"_LANGUAGES_"に格納されます。

設定オプション"ok_languages"で受け取ってもよい言語の言語コードをスペース区切りで並べて記述します。言語コードは 2 文字のもので小文字で記述します。例えば、英語と日本語だけを受け取ってもよい場合は次のように記述します。

```
ok_languages en ja
```

なお、デフォルトでは全ての言語を許容する"all"が設定されています。詳しくは Mail::SpamAssassin::Plugin::TextCat のマニュアル(POD)を参照してください。

このプラグインの処理には時間がかかるので有効にする際には注意してください。

基本設定

v310.pre において次の行を有効にしてください。

```
loadplugin Mail::SpamAssassin::Plugin::TextCat
```

ルールの記述例

```
ifplugin Mail::SpamAssassin::Plugin::TextCat

### 英語と日本語だけが欲しい場合
ok_languages en ja

### scoreの再割り当て
#    判定ミスの場合の影響が大きいのでスコアを小さくする
score UNWANTED_LANGUAGE_BODY    1.0

#    Shift_JISを許容するためスコアを0にする
score BODY_8BITS                0.0

### メッセージにヘッダを追加する場合
#    追加するヘッダの例) X-Spam-Language: ja.iso-2022-jp
add_header all Language _LANGUAGES_

endif # Mail::SpamAssassin::Plugin::TextCat
```

ネットワークテスト関連

DCC

説明

DCC(Distributed Checksum Clearinghouse)はスパムメールのチェックサムを集め、スパムの検出を行うシステムです。このプラグインはこの DCC をスパムの検出に利用します。

DCC については詳しくは次のサイトをご覧ください。

http://www.rhyolite.com/anti-spam/dcc/

DCC はオープンソースではないため、このプラグインはデフォルトで無効になっています。使用するにあたっては上記サイトのライセンスを参照してください。

必要なソフトウェア

先のサイトで配布している DCC をインストールする必要があります。

基本設定

v310.pre において次の行を有効にしてください。

```
loadplugin Mail::SpamAssassin::Plugin::DCC
```

ルールの記述例

デフォルトのルールで利用できます。

Pyzor

説明

Pyzor はスパムを検出してブロックするための協調ネットワークシステムです。このプラグインは Pyzor をスパムの検出に利用します。

Pyzor については詳しくは次のサイトをご覧ください。

http://pyzor.sourceforge.net/

必要なソフトウェア

先のサイトで配布している pyzor をインストールする必要があります。

基本設定

インストール時に有効になっているはずですが、v310.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::Pyzor
```

ルールの記述例

デフォルトのルールで利用できます。

Razor2

説明

Vipul's Razor はユーザによるスパムの報告に基づいたスパム検出サービスです。このプラグインはこの Razor をスパムの検出に利用します。また、スパムを報告します。

Razor については詳しくは次のサイトをご覧ください。

http://razor.sourceforge.net/

必要なモジュール

Razor2::Client::Agent

このモジュールは次のサイトから取得できます。

http://razor.sourceforge.net/

基本設定

インストール時に有効になっているはずですが、v310.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::Razor2
```

ルールの記述例

デフォルトのルールで利用できます。

SpamCop

説明

SpamCop はスパムを報告するサービスです。SpamCop はスパムメールの発信元を調べ、ISP に報告します。このプラグインは SpamCop へスパムを報告します。

SpamCop については詳しくは次のサイトをご覧ください。

http://www.spamcop.net/

必要なモジュール

Net::DNS

Net::SMTP

基本設定

インストール時に有効になっているはずですが、v310.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::SpamCop
```

ルールの記述例

SpamCop のサイトの説明をよく読んだ上で設定してください。

```
ifplugin Mail::SpamAssassin::Plugin::SpamCop

# あなたのメールアドレス
spamcop_from_address foo@example.org

# SpamCopへの投稿先メールアドレス
spamcop_to_address spamassassin-submit@spam.spamcop.net

endif #  Mail::SpamAssassin::Plugin::SpamCop
```

URIDNSBL

設定

このプラグインはメッセージの本文に記述された URI のドメイン名をブラックリストから検索します。検索結果はデフォルトのルールにより評価され、スコアが付きます。

基本設定

インストール時に有効になっているはずですが、init.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::URIDNSBL
```

ルールの記述例

デフォルトのルールで利用できます。

送信者認証関連

送信者認証技術は送信元が詐称されていないことの保証でしかなく、スパムであるかどうかの判断ではありません。この点に注意してください。

SPF

説明

このプラグインは SPF(Sender Policy Framework)の検証を行います。

必要なモジュール

Mail::SPF::Query

基本設定

インストール時に有効になっているはずですが、init.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::SPF
```

ルールの記述例

デフォルトのルールで利用できます。

送信者のドメインの SPF の設定が間違っているけど直らないなどによりホワイトリストに追加したい場合は次のようなルールを記述してください。

```
ifplugin Mail::SpamAssassin::Plugin::SPF

whitelist_from_spf joe@example.com fred@example.com
whitelist_from_spf *@example.com

endif    # Mail::SpamAssassin::Plugin::SPF
```

DomainKeys

説明

DomainKeys は電子署名を利用する送信ドメイン認証です。

このプラグインは DomainKeys の検証を行います。DomainKeys 自体が実験的なものなので本格的には利用しないでください。

また、メールマガジンやメーリングリストにおいてヘッダが挿入されたために署名の検証に失敗することがあるので注意してください。

必要なモジュール

Mail::DomainKeys

基本設定

v310.pre において次の行を有効にしてください。

```
loadplugin Mail::SpamAssassin::Plugin::DomainKeys
```

ルールの記述例

```
ifplugin Mail::SpamAssassin::Plugin::DomainKeys

### ポリシーが"sign all email"であるが、署名が無い場合
meta DK_UNSIGNED        (!DK_SIGNED) && DK_POLICY_SIGNALL
describe DK_UNSIGNED    Domain Keys: message does not have any
signature
score DK_UNSIGNED       1.0

### ポリシーが"sign some email"であるが、署名が無い場合
meta DK_UNSIGNED2       (!DK_SIGNED) && DK_POLICY_SIGNSOME
describe DK_UNSIGNED2   Domain Keys: message does not have any
signature
score DK_UNSIGNED2      0.1

### 署名があるが、検証に失敗した場合
meta DK_UNVERIFIED      (!DK_VERIFIED) && DK_SIGNED
describe DK_UNVERIFIED  Domain Keys: a signature of message is
unverified
score DK_UNVERIFIED     0.5

### 署名が付いているはずのドメインから来たメッセージで電子署名
が
### 付いていない場合
# yahooから来たメッセージで電子署名がない場合
header __FROM_YAHOO          From =~ /[¥@¥.]yahoo¥./
meta DK_UNSIGNED_YAHOO       !DK_SIGNED && __FROM_YAHOO
describe DK_UNSIGNED_YAHOO   Domain Keys: unsigned yahoo domain
score DK_UNSIGNED_YAHOO      2.0
```

```
# gmailから来たメッセージで電子署名がない場合
header __FROM_GMAIL          From =~ /¥@gmail¥.com/
meta DK_UNSIGNED_GMAIL       !DK_SIGNED && __FROM_GMAIL
describe DK_UNSIGNED_GMAIL   Domain Keys: unsigned gmail.com
score DK_UNSIGNED_GMAIL      2.0

endif # Mail::SpamAssassin::Plugin::DomainKeys
```

DKIM

説明

DKIM(DomainKeys Identified Mail)は電子署名を利用する送信ドメイン認証です。

このプラグインは DKIM の検証を行います。DKIM 自体が実験的なものなので本格的には利用しないでください。

必要なモジュール

Mail::DKIM

Crypt::OpenSSL::Bignum

基本設定

v312.pre において次の行を有効にしてください。

```
loadplugin Mail::SpamAssassin::Plugin::DKIM
```

ルールの記述例

```
ifplugin Mail::SpamAssassin::Plugin::DKIM

### ポリシーが"sign all email"であるが、署名が無い場合
meta DKIM_UNSIGNED       (!DKIM_SIGNED) && DKIM_POLICY_SIGNALL
describe DKIM_UNSIGNED   Domain Keys: message does not have any
signature
score DKIM_UNSIGNED      1.0

### ポリシーが"sign some email"であるが、署名が無い場合
meta DKIM_UNSIGNED2      (!DKIM_SIGNED) && DKIM_POLICY_SIGNSOME
describe DKIM_UNSIGNED2 Domain Keys: message does not have any
signature
score DKIM_UNSIGNED2     0.5

### 署名があるが、検証に失敗した場合
meta DKIM_UNVERIFIED     (!DKIM_VERIFIED) && DKIM_SIGNED
describe DKIM_UNVERIFIED Domain Keys: a signature of message
is unverified
score DKIM_UNVERIFIED    0.5

endif # Mail::SpamAssassin::Plugin::DKIMIM
```

Hashcash

説明

hashcash は DoS ベースのスパム対策技術の一つです。hashcash の基本的な考え方はメッセージの送信者に送信のコスト（ハッシュ値の計算による CPU 処理時間）を負担してもらおうというものです。

hashcash について詳しくは次のサイトをご覧ください。

http://www.hashcash.org/

http://en.wikipedia.org/wiki/Hashcash

このプラグインはこの hashcash の検証を行います。

基本設定

インストール時に有効になっているはずですが、init.pre において次の行が有効になっていることを確認してください。

```
loadplugin Mail::SpamAssassin::Plugin::Hashcash
```

ルールの記述例

デフォルトのルールで必要な設定が行われているので、"hashcash_accept"オプションでメールアドレスを登録するだけで使えます。メールアドレスにはファイルグロブのように"*"と"?"が使えます。

```
ifplugin Mail::SpamAssassin::Plugin::Hashcash

# 受信者がfoo@example.orgの場合
hashcash_accept foo@example.org

# サイトのドメインがexample.orgの場合
hashcash_accept *@example.org

endif # Mail::SpamAssassin::Plugin::Hashcash
```

その他

AccessDB

説明

Sendmail や Postfix のような MTA はアクセス制御を行うための"access"データベースを持っています。このプラグインはその"access"データベースを SpamAssassin から利用できるようにしたものです。

エントリが見つからない場合あるいはアクションが"OK"か"SKIP"の場合は false を返します。エントリが存在し、アクションが"REJECT"か"ERROR"か"DISCARD"の場合は true を返します。

基本設定

v310.pre において次の行を有効にしてください。

```
loadplugin Mail::SpamAssassin::Plugin::AccessDB
```

ルールの記述例

デフォルトのルールで利用できます。

"access" データベースの場所をデフォルトの "/etc/mail/access.db"から他の場所に変えたい場合は次のように記述します。設定ファイルの読み込みの順番に気をつけてください。

```
ifplugin Mail::SpamAssassin::Plugin::AccessDB

header ACCESSDB eval:check_access_database('/etc/access.db')

endif # Mail::SpamAssassin::Plugin::AccessDB
```

AntiVirus

説明

このプラグインは簡易なウィルスチェックを行います。

実際はウィルスを検出するわけではなく、次のような疑わしい添付ファイルを検出をします。

- 実行可能なタイプの拡張子(com, exe, pif, scr 等)が付いているファイル
- base64 にエンコードされた実行ファイル
- uuencode された実行ファイル
- ファイル名の拡張子と Content-Type から疑わしいと判断されたファイル

そのため、ウィルスを確実に検出したい場合は ClamAV のようなアンチウィルスのソフトウェアを使用してください。

基本設定

v310.pre において次の行を有効にしてください。

```
loadplugin Mail::SpamAssassin::Plugin::AntiVirus
```

ルールの記述例

デフォルトのルールで利用できます。

サードパーティのプラグイン

SpamAssassin はプラグインで機能を拡張できるため、サードパーティのプラグインの開発も行われています。次のサイトでサードパーティのプラグインが紹介されています。

http://wiki.apache.org/spamassassin/CustomPlugins

この中で非常に興味深い FuzzyOcrPlugin をここで紹介します。

FuzzyOcrPlugin

説明

最近はボディの文章が少なめで、画像に表示されるのが文章である画像ファイル付きのスパムメールをよく見かけるようになりました。このようなメールはボディの文章が少ないので文章の解析だけではスパムであるかどうかの判断が難しいです。そこで、画像を OCR プログラムで解析して単語を抽出して判断するプラグインがいくつか開発されました。ここで紹介する FuzzyOcrPlugin もその中の一つです。

FuzzyOcrPlugin について詳しくは次のサイトをご覧ください。

http://wiki.apache.org/spamassassin/FuzzyOcrPlugin

なお、このプラグインの解析処理には数秒の時間がかかるので、処理時間を踏まえて利用してください。

インストール

次のサイトから FuzzyOcr の tar ball をダウンロードしてください。

http://users.own-hero.net/~decoder/fuzzyocr/

ダウンロードしたファイルを展開して、付属の文書の指示に従いインストールを行ってください。

設定に関する注意事項

デフォルトの設定では"focr_logfile"と"focr_digest_db"で指定するファイルにパーミッションに問題が生じるものがあります。適切な場所およびパーミッションを設定してください。

OCR エンジンによる解析処理に時間がかかるため、不必要な解析処理を回避する機能があります。

このプラグインのテストを行う前に他のテストで十分に高いスコアが付いていればこのテストを行うまでもありません。その判断基準となるスコアを設定するオプション"focr_autodisable_score"が用意されています。このオプションで指定したスコア（デフォルトでは 10）以上であれば解析処理を行いません。必要であれば、"required_score"の設定値などのバランスを考慮した上でこの設定値を変更してください。

また、実験的な機能として、一度解析した画像のハッシュを記録しておき、次回以降にそれを再利用して解析処理を回避する機能があります。デフォルトで無効になっているので"focr_enable_image_hashing"を"1"に設定することで利用することが出来ます。

SpamAssassin のプラグイン紹介

発行日　2006 年 10 月 28 日

著　者　滝澤 隆史