Vol.2010-IOT-8 No.12
2010/3/1

メールサーバにおける多様な迷惑メール対策の統合管理手法

佐々木　琢磨[†1]　阪口哲男[†1]

迷惑メールは, メールサーバに負荷になるだけではなく, その対策で管理者に負担をかけている. そのような管理の負担を低減すべく, 本研究ではメールサーバにおける多様な迷惑メール対策を統合して管理する手法を開発した. 既存の迷惑メール対策の分析結果より, その設定記述の統一のための言語を, 電子メールフィルタリング用の言語 Sieve に基づいて定義した. この言語を用いることで, メールサーバ全体を統合して管理する. 言語解釈系は, 多様な迷惑メール対策に柔軟に対応可能な開かれた構成方式で実現した. 具体的には, 迷惑メール対策ソフトウェア等の設定テンプレートと変換ルール記述を与えることで, 各ソフトウェアの設定ファイルの変換を可能とした. 本手法を適用したメールサーバの試験運用結果より, 本手法の利点について論じる.

An integrated method for management of mail server with various anti-spam tools

Takuma SASAKI[†1] and Tetsuo SAKAGUCHI[†1]

Bulk spam mails are heavy loads for mail servers and administrators of mail servers spend much time on protecting their servers from spam mails. To solve this problem, this paper describes development of an integrated method for management of mail servers with various anti-spam tools. We define a unified language based on the language Sieve for writing configurations of mail server softwares and anti-spam softwares. It has the notation based on abstraction from some existing mail server softwares and anti-spam softwares. We also define a method of construction of a translator of the language. The translator uses templates of configuration files of the softwares and descriptions of rules on converting parameters, when producing the configuration files from a unified configuration written in the language. This paper also discusses the advantages of the method through administrative operations of an experimental mail server with the proposed method.

1. はじめに

インターネットにおける迷惑メールは, ユーザへの被害だけではなく, 同時多量送信によるメールサーバへの負荷も問題であり, メールサーバへの迷惑メール対策の導入は必須である. 一方で, 実際に利用されているメールサーバソフトウェアは様々であり, 迷惑メール対策は, 同じ原理や方式のものでも, メールサーバソフトウェアごとに個別にある. 様々なソフトウェアを組み合わせるために, 設定ファイルが散在していたり, 個々の設定ファイルが固有の記述形式であるなど, 管理者に負担がかかるという問題がある. その問題の解決のため, 本研究ではメールサーバの構成要素と迷惑メール対策手法を分析し, 各要素と迷惑メール対策の統合管理手法とその実現方法を提案する.

2. メールサーバと迷惑メール対策の現状

本研究では, MSA, MTA, MDA, MRA からなるメール配送システム[1] をメールサーバとする. また, MSA, MTA, MDA, MRA それぞれをメールサーバ構成要素と呼ぶ.

メールサーバにおける迷惑メール対策は, 初めて接続してきた送信要求に対して一時エラーなどの遅延対応をし, メールの再送要求を待つ, Evan Harris の Greylisting[3] という方式が代表的である. Greylisting は MTA に導入されるが, 主に MDA や MUA において, 宛先ユーザが指定した迷惑メール中の文字列パターンを利用するルールベース方式, 多数のユーザを抱えるメールサーバの MDA において, 各ユーザの迷惑メール判定情報を共有して迷惑メール判定に用いる協調フィルタリング方式がある.[2]

メールサーバにおける迷惑メール判定で利用できる迷惑メールの特徴は, メールサーバ構成要素それぞれにおいて異なるので, 手法によって組み入れ可能な構成要素が異なる.

このように, メールサーバにおける迷惑メール対策は, 様々な方式がいくつかあり, それらを組み合わせることによってより有効な迷惑メール対策ができることがこれまでの研究で明らかになっている.[4),5]

また, 実際の運用においても, 組み合わせによる有用性が示されており, サーバ管理・運用者向けの雑誌において, 迷惑メール対策の現状および導入手順・管理について特集されるこ

†1 筑波大学大学院図書館情報メディア研究科
Graduate School of Library, Information and Media Studies, University of Tsukuba

ともある.[6),7)]

迷惑メール対策の原理が同じであっても, メールサーバソフトウェアごとに導入するソフトウェアや設定方式が異なることから, メールサーバの管理・運用手順はそれらの組み合わせにより様々となるため, 管理者はその選定から個々のソフトウェアの設定方法, 組み合わせ方まで熟知する必要がある. さらに, 複数の迷惑メール対策の原理・方式のフィルタソフトウェアを併用し, 全体として有効な迷惑メール対策を構成しようとする場合, 設定した時点で効果的であっても, 特徴が頻繁に遷移する迷惑メールに対して効果を挙げ続けるのは難しく, 柔軟に設定を管理していく必要がある.

そこで本研究では, 多様な迷惑メール対策をメールサーバと統合して管理することを可能とし, このような管理者の負担を低減する手法を考案する.

3. 迷惑メール対策の統合管理

複数の迷惑メール対策をメールサーバで併用する場合, メールサーバソフトウェアと, 迷惑メール対策のために導入したフィルタソフトウェアの両方を設定・管理する必要がある. しかし, メールサーバソフトウェアは迷惑メール対策設定以外にも設定事項はあり, 場合によっては基本的なメールサーバの動作設定と迷惑メール対策設定が同一の設定ファイル内に混在することもある. また, フィルタソフトウェアを複数導入すると, 各フィルタソフトウェアの設定・管理も必要となる.

その結果, 迷惑メール対策を含めたメールサーバ全体の設定内容を確認する場合, 散在した各設定ファイルを辿っていく必要があり, このような作業はメールサーバ管理者にとっては大きな負担となる. そこで, 迷惑メール対策の統合管理が必要と考えられる. 迷惑メール対策の統合管理とは, メールサーバソフトウェアの設定と迷惑メール対策設定の記述や記法の統合を目的としたメールサーバの管理方式である.

迷惑メール対策の統合管理は, いくつかの手法がこれまでに研究・実用化されてきた. ここでは, MTA ソフトウェアで利用される外部ソフトウェアの規格統一のための API である milter API および milter manager と, 電子メールのフィルタリング動作を記述できるプログラミング言語 Sieve を取り上げる.

(1) milter API と milter manager

milter API は, MTA ソフトウェアの sendmail が, sendmail に適用する外部の迷惑メール対策ソフトウェアの規格を統一するための API として導入したものである. これによって, MTA における外部ソフトウェアの呼び出し方や, MTA における SMTP セッションやメール本文などの情報を外部のソフトウェアで利用する仕組みが統一化された. これによって, milter API に従った多くの milter アプリケーションが開発・提供されている.

milter API は, postfix を始めとする他のメールサーバソフトウェアでも導入されている. これにより, 様々な迷惑メール対策方式が milter として開発され, MTA における異なった迷惑メール対策方式の併用および導入が容易となった.

また, MTA に導入されている複数の milter 管理を目的とした milter manager[8)] も開発された. milter manager は, MTA によって唯一の milter として呼び出され, さらに他の milter の制御が可能なため, 各 milter アプリケーションの呼び出し条件の管理や milter アプリケーション間における共通パラメータの記述が可能となった.

しかし, メールサーバにおける迷惑メール対策の統合管理の観点では, milter は MTA 以外に導入される迷惑メール対策は対象とならない.

(2) Sieve

Sieve は電子メールフィルタリングのためのプログラミング言語であり, その基本仕様は RFC 5228[9)] に示されている. 従来のフィルタソフトウェアでは, パターンや条件分岐の設定記述は, それぞれのソフトウェア固有の記述が必要であった. Sieve によって, そのような固有の記述についての統一化の方向性が示された. Sieve は, その基本仕様において記述できることはそれほど多くはないが, メールの振り分け条件に関して汎用性の高い記述が可能な文法となっている.

しかし, Sieve およびその拡張は記述形式が規定されているだけのことが多く, メールサーバにおける Sieve 記述の具体的な処理方式までは規定されていない. そのため, Sieve が利用できるメールサーバソフトウェアであっても, 利用できる Sieve 記述は一部の機能に限定されていることが多い.

本節では milter API と Sieve という迷惑メール対策統合の仕組みを取り上げたが, 前者は MTA の迷惑メール対策ソフトウェアの導入や設定手順の統合, 後者はメール振り分け条件記述の統一にとどまっている. また, それぞれ MTA や MDA における迷惑メール対策の統合と限定的である. そこで, 本研究ではメールサーバ構成要素全体を対象とし, 迷惑メール対策の統合管理のための記述言語を提案する.

表 1 milter-greylist を導入する際の MTA による設定の差異

MTA	設定項目	設定するファイル
sendmail	INPUT_MAIL_FILTER の値変更	sendmail.cf
postfix	smtpd_milters の値変更	main.cf

表 2 postfix に Greylisting を実現する際のソフトウェアによる設定の差異

フィルタソフト	再送禁止時間指定例	リスト保持時間指定例	設定するファイル
milter-greylist	delay 5m	autowhite 3d	greylist.conf
postgrey	–delay=300	–max-age=3	/etc/init.d/postgrey

4. メールサーバ統合管理言語

4.1 従来手法の問題と統合管理言語の提案

既に述べたように, メールサーバソフトウェアとフィルタソフトウェアの組み合わせは様々である. 例えば, postfix に Greylisting を適用する場合, postgrey を導入することも, postfix が milter API に対応していることから, milter-greylist を導入することもできる. このように, 同じ原理や方式に基づく迷惑メール対策を導入する場合でも, 導入する外部ソフトウェアの違いによって設定項目や管理方式が異なってしまう場合がある. また, milter-greylist を利用する場合であっても, sendmail と postfix では MTA 自体の設定記法が異なるため, 実際のメールサーバ管理者以外がメールサーバ全体の構成を把握することが難しく, サーバ運用上の問題対応にかかるメールサーバ管理者の負担が大きくなる. これらの違いの例を表 1, 2 に示す.

そこで, 迷惑メール対策およびそれを導入するのに必要な設定項目の記述形式を統一したメールサーバ統合管理言語を定義し, その記述から各フィルタソフトウェアの設定ファイルおよびメールサーバ構成要素の設定ファイルを生成する解釈系を開発した. これらの言語およびその解釈系を適用したメールサーバは, メールサーバ全体の構成を確認する場合に散在した設定ファイルを辿ることがなくなり, メールサーバ管理の負担が低減される.

4.2 統合管理言語 amaretto

本研究で開発したメールサーバ統合管理言語を amaretto と呼ぶ. 図 1 に amaretto の記述例を示す. 図 1 は, MTA に postfix を用い, 迷惑メール対策として postgrey を導入して Greylisting を行う場合の amaretto の記述例である. また, 出力結果となる postgrey および postfix の設定ファイルの一部を抜粋し, 図 2, 3 に示す.

```
 1  require "amaretto";
 2    set "greyport" "60000";
 3
 4    use "postgrey" as GREYLISTING{
 5      delay "300";
 6      postgrey.port "${greyport}";
 7    };
 8
 9    use "postfix" as MTA{
10      service "${greyport}";
11    };
```

図 1 amaretto の記述例

```
#!  /bin/sh
set -e
PATH=/sbin:/bin:/usr/sbin:/usr/bin
...
POSTGREY_OPTS="--delay = 300 --inet= 60000"
...
```

図 2 図 1 から出力される postgrey 起動スクリプト（抜粋）

図 1 を用いて, amaretto 記述の概要を説明する. amaretto は Sieve の文法に基づいて定義している. Sieve は基本文法が単純であり, 元々電子メールフィルタリングに使われる言語であるため, 本手法の焦点である迷惑メール対策記述の統合を実現するのに適していると考えた.

1 行目における `require` 文は, amaretto 記述であることを明示する部分である.

2 行目の `set` 文は, 既存の Sieve の変数の拡張機能を用いている. 変数の概念を統合記述言語に取り入れることにより, 複数の構成要素に共通するパラメータをまとめて指定することが可能となる. この例では, 変数 `greyport` に値 `60000` を格納している. `60000` は postgrey が動作するポート番号である.

```
myhostname = utena.slis.tsukuba.ac.jp
mydomain = utena.slis.tsukuba.ac.jp
...
check_policy_service   inet:60000
...
```

図 3 図 1 から出力される postfix の main.cf ファイル（抜粋）

4-7 行目, 9-11 行目が構成要素ごとの設定記述部分である．amaretto 記述では, 迷惑メール対策方式やメールサーバ構成要素の名称と, 使用する具体的なソフトウェア名を use ブロックにより指定する．

5 行目における delay "300"; は, 迷惑メール対策方式として Greylisting を用いる際の再接続と見なすための時間の指定である．このパラメータの指定は, GREYLISTING 方式を記述内で扱う場合に最低限必要なものである．Greylisting を実現する外部のソフトウェアが変わっても, このパラメータは Greylisting に必要なものであり, 例えば"postgrey"が"milter-greylist"に変更された場合であっても, 変更の必要はない．

6 行目の postgrey.port は Greylisting 方式共通のパラメータではなく, postgrey が独自に機能として持つパラメータ port を指定するための書式である．このように, Greylisting 方式共通のパラメータである delay の設定記法と, Greylisting を実現するソフトウェア固有のパラメータ port の指定記法を分けることにより, 迷惑メール対策がどのように設定されているかの判別が容易となる．また, 既に設定してある変数の参照は Sieve の変数拡張に基づいており, ${greyport} のように記述する．図 2 は, 以上の Greylisting 設定記述によって出力される postgrey の起動スクリプトである．スクリプト内に出現する 300, 60000 が解釈系において amaretto 記述の対応部分と置き換えらた部分である．

10 行目は, postfix の設定ファイルに postgrey が動作するポート番号を指定することが必要となるため, service に postgrey が動作するポート番号が格納されている変数 greyport を指定したものである．なお, メールサーバ構成要素のソフトウェアは, 迷惑メール対策ソフトウェアと異なりその設定事項にソフトウェア固有の部分が多く, 記述全体が冗長となることを回避するため, 6 行目のように接頭辞として postfix. は付与しないものとした．この設定記述から出力された postfix の設定ファイル (main.cf) が図 3 である．図 2 と同様に, 60000 の部分が解釈系において置換処理がされた部分である．

```
1   delay %{delay}m
2   autowhite %{max-age}d
```

図 4 表 2 の milter-greylist 用テンプレート（抜粋）

```
1   using GREYLISTING
2   if delay :exists
3     then value.to_i / 60
```

図 5 milter-greylist のルール記述の例

5. 多様な迷惑メール対策に柔軟に対応可能な解釈系の構成方式

本節では, amaretto 記述を多様に存在する迷惑メール対策に対して柔軟に対応させるための方式について述べる．amaretto の解釈系では, 各ソフトウェアの設定ファイルのテンプレートに対し amaretto 記述の値を適用することにより設定ファイル生成を実現する．

ここで, amaretto の Greylisting 設定において, 再送禁止時間およびリスト保持時間の項目名がそれぞれ, delay, max-age で指定されているとして, 表 2 の設定で milter-greylist を導入する場合の milter-greylist のテンプレートを図 4 に示す．テンプレートにおける amaretto 記述内の値を参照する場合の書式は, 1,2 行目のように%および {} を用いて記述する．また, milter-greylist において時間の指定には, 時間の単位を示す接尾辞の"m"や"d"が必要であることから, 図 4 のようにそれらを補った記述となる．しかし, amaretto 記述内の delay の値が秒指定されていた場合, テンプレートにそのまま反映させることはできない．

このような場合のための変換規則をルール記述としてテンプレートと共に用いる．ルール記述の例は図 5 のようになる．1 行目は, Greylisting 方式の迷惑メール対策ソフトウェアの重複確認を記述している．ルール記述を用いることにより, このような簡単な変換や制約を記述することが可能となる．2 行目は, amaretto 記述内の delay について, 簡単な計算処理を行っている．これは, amaretto 記述の delay の秒指定を分に変換する記述例である．テンプレートおよびルール記述の導入により, 多様な迷惑メール対策に柔軟に対応させていくことが可能である．解釈系の構成を図 6 に示す．

6. 解釈系の実現

本研究における解釈系は前処理として amaretto 記述とルール記述を入力し, ルールを適

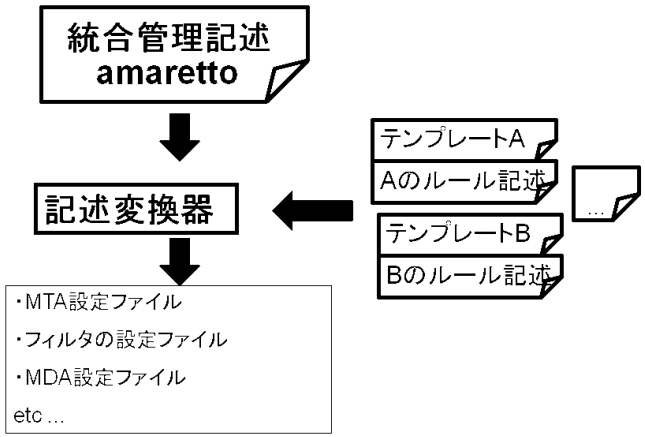

図 6 解釈系の構成

用した中間ファイルを出力するルール解釈部分と, その中間ファイルとテンプレートを入力して該当部分を置換処理する amaretto 記述解釈部分に大きく分けられる.

解釈系の開発は, Java 言語を用いた. parser generator には SableCC[10] を用い, 生成された各クラスを利用して開発した. 入力された amaretto 記述は, 適用ルール記述を解釈する部分により記述の計算や制約の確認がされ, amaretto 記述解釈部分に渡される. amaretto 記述の処理系も, 適用ルール記述の処理系と同様に SableCC を用いて開発を行った.

7. 試験運用と評価

本研究のメールサーバ統合管理手法を適用したメールサーバを試験運用し, 迷惑メール対策の設定・更新作業を行うことにより, 本手法における管理の手間を従来の場合と比較して評価を行った.

試験運用するメールサーバは, VMWare 仮想サーバ上に Debian GNU/Linux 5.0 をゲスト OS として, MTA ソフトウェアとして Postfix 2.5, MDA・MRA ソフトウェアとして Cyrus IMAP Server 2.2 を用いて行った. また, 導入する迷惑メール対策方式としては, Greylisting, Whitelist, Cyrus IMAP Server がサポートする Sieve 機能を採用した. また, Greylisting に関しては, その方式を実現するためのソフトウェアとして postgrey および milter-greylist を採用した.

(1) Greylisting 実現ソフトウェアの変更

まず, 導入済みの Greylisting ソフトウェアを postgrey とする. それを milter-greylist に変更し, Greylisting 方式としての再送禁止時間および正当サーバリスト保持時間, さらに, 再送と見なさない最初の接続からの経過時間の設定をそれぞれ 10 分, 30 日, 1 週間と設定する場合の作業を行った. その作業内容は, 従来は以下のようになる.

- postfix の設定ファイル操作 (postgrey 設定の削除および milter-greylist 設定の追加)
- milter-greylist の設定ファイルに各設定内容を記述

これに対し, 本手法では上記のうち, postfix の設定ファイルと milter-greylist の設定ファイルは 1 つの amaretto 記述で統合管理されているので, 直接各設定ファイルを変更する手間が低減される.

一方, milter-greylist の設定に関しては, 従来は, milter-greylist のマニュアルなどに従って, 各パラメータの項目名や値の書式などを記述する必要があったが, 本手法では, 設定ファイルのテンプレートおよびルール記述が準備されているので, amaretto 記述のみで操作を完結できる.

(2) Greylisting 実現ソフトウェアの変更

従来は, postfix における該当行を変更する. その上で, postgrey の起動スクリプトの記述マニュアルに従い, 各パラメータの設定を行う. ここで, milter-greylist との時間設定の書式の違いを留意して再設定することになる.

一方, 本手法を用いて同作業を行う場合, postgrey と milter-greylist 間で Greylisting に必要な設定 (`delay, max-age` など) 項目の記述や記法は統合され, かつ, 時間の接尾辞に関しても, ルール記述により追加されるので, amaretto 記述の変更だけで設定が完結する.

このように, 特定の迷惑メール対策方式において, その実現ソフトウェアの変更作業を行う場合, 本手法は従来に比べ手間の低減が見込まれる.

(3) 迷惑メール対策の追加

上記設定のメールサーバに, Whitelist, Sieve の設定をする.

従来手法は, これまでと同様に, 各ソフトウェアのマニュアルに従い, 設定パラメータ名や値の書式を記述する必要があるため, 最低でも操作対象は 4 ファイル計 300 行程度になる. 一方, 本手法では 30 行程度の amaretto 記述のみで作業が完結する.

このように, 導入ソフトウェアが増えても, それぞれの固有の記法を意識せずに, 統一された amaretto 記述で各パラメータの設定ができることは, 管理者の作業の手間が低減すると考えられる.

(4) 全体の評価と展望

本手法を適用すると, 迷惑メール対策方式自体は変更せず, それが実現されるソフトウェアを変更する作業を行った際に, 手間の低減が見込まれる. また, 単一の amaretto 記述を参照することで, 散在した各設定ファイルを参照する作業が不要となり, 容易にメールサーバ全体の構成を確認することができ, 配送ミスなどの問題発生に対応する際の管理者の手間の大幅な低減が可能となる.

また, 各迷惑メール対策方式における必須パラメータおよび amaretto 記述におけるパラメータ名, その値の書式などにに関しては, 限られた迷惑メール対策やその実現ソフトウェアを分析した上で定めたものに過ぎないため, 本手法を普及させる上ではより広くの情報や根拠が必要となると考えられる. そして, 各ソフトウェアのテンプレート作成が広まることで, 本手法によるメールサーバの統合管理の有用性がより高まっていくと考えられる.

8. おわりに

メールサーバ統合管理言語 amaretto とその解釈系を用いた本手法は, メールサーバの設定・更新作業において, 従来よりも管理者の作業の手間が低減されることが示された.

また, 今後まったく新しい迷惑メール対策方式が実現された場合でも, その方式のテンプレートおよびルール記述を用意することで本手法は対応できるので, 有用性は高いと考えられる.

参考文献

1) David Wood. “インターネット上の電子メール”. 電子メールプロトコル. 佐々木雅之ほか監訳. オライリー・ジャパン, 2002, p. 1-24.
2) 渡部綾太, 愛甲健二.“スパム防衛”. スパムメールの教科書. データハウス, 2006, p. 79-282.
3) Evan Harris. The next step in the spam control war : Greylisting. 2003. http://projects.puremagic.com/greylisting/whitepaper.html
4) 小池, 佐藤. ユーザ・プロバイダ連携によるスパムメール・フィルタリングの検討. 信学技報, ISEC2007-13, pp85-92, 2007.
5) 陳, 佐々木, 田中. SMTP セッションフィルタとグレイリストを併用した迷惑メール対策. 情報処理学会論文誌, Vol47, No4, pp1000-1008, 2006.
6) 佐藤潔. “Postfix で spam メール対策を実現”. Software Design. 2008, 3, p. 38-64.
7) 須藤功平. “迷惑メール対策のいま”. セキュリティExpert. 2009, p. 100-103.
8) クリアコード. milter manager. http://milter-manager.sourceforge.net/index.html
9) RFC 5228. Sieve: An Email Filtering Language. http://tools.ietf.org/rfc/rfc5228.txt
10) SableCC project. SableCC. http://sablecc.org/wiki