

SMTP セッションでの SPAM 受信拒否

広瀬雄二

概 要

近年電子メールという手段を用いて、不要な広告や勧誘文を大量無差別に送りつけるUBE(Unsolicited Bulk E-mail)、いわゆる SPAM が社会問題となっている。SPAM にはそれ特有のパターンが本文に含まれるため、それらを統計的に処理することで自動的に排斥する「SPAMフィルタ」が発達している。そのいっぽうで、受信サーバに届く SPAM の数が増大し、SPAM フィルタを動かすためのサーバ負荷が無視できなくなってきたのも事実である。本論では、SPAM 送信者が「身元を詐称する」という性質を利用し、SPAM のメッセージ本体を受け取る前段階で送信自体を門前払いする方式を提案・実装し、評価する。

SPAM rejection at SMTP session

HIROSE, Yuuji

Tohoku University of Community Service and Science

Abstract

UBE(Unsolicited Bulk E-mail), so called "SPAM", had become the social problem in the Internet community. Filtering out SPAMs by analyzing the pattern of their message body became recent hot interest of researchers, so that many SPAM-Filtering systems are in practical use. However, it had become problem that too many SPAM messages raise load of SMTP servers.

In this paper, we designed, implemented and evaluated the method of SMTP rejection at SMTP session before receiving whole message of SPAM.

東北公益文科大学 yuuji@koeki-u.ac.jp

1 背景

電子メールは送受信にかかるコストが低い上に短時間で相手に届くなどメリットが多く、現代には欠かせない通信手段となっている。しかしそれを悪用し、望まないものにまで広告を無差別に大量送信する悪徳業者があつと絶たない。大量に送られる望まれない電子メールのことを UBE¹ といい、一般的には SPAM という呼称で知られている。

近年激増した SPAM により、本来読みたいと望んでいるメッセージが埋もれてしまうことが問題化してきた。これを解決するため受信したメッセージの本文を統計的手法により自動検定し、SPAM である可能性の高いものを除外するシステムが求められている。このようなシステム「SPAM フィルタ」は、より高精度な選別ができるような研究が多くなされている。

ただし SPAM フィルタ自身高度な推定処理を行なうため、計算量は少なくない。これをメールサーバに到着するすべてのメッセージに対して施すことを考えた場合の計算量は膨大になる。本来は無用なものである SPAM を判定するための処理が、貴重な計算機資源を消費しすぎることも顕在化してきている。

またいっぽうでは、不特定多数のものが悪用する可能性のあるメールサーバの IP アドレスを集めてブラックリストとしたデータベースを集中的に作り、世界各地から利用する試みもある。このような、送信メールサーバのブラックリストデータベースは SPAM フィルタの登場以前より数多く開発・活用されている。MAPS[2] や SORBS[3] は代表的なものであるが、それらは欧米主体の判断基準でデータベースに登録するため、たとえばアジア圏に割り当てられた IP アドレス群が、無条件で必要以上の大きさでブラックリストに登録されてしまうといった問題が起きている。

本論では、SPAM を発信する身元という情報を元にメッセージを受け取る前に SPAM の可能性の高いものを、地域性を考慮して門前払いとする方式を提案し、実際に設計、構築、運用した結果について考察する。

¹Unsolicited Bulk E-mail

2 SMTP セッションの 3 要素

電子メールをサーバどうしがやりとりするときの送信手順の取り決めを SMTP(Simple Mail Transfer Protocol) といい、送信側のサーバは受信側のサーバに対する電子メールの送信に先立ち、“HELO”, “MAIL FROM”, “RCPT TO” の 3 つの情報をこの順で送らなければならない(図1)。この 3 要素は以下の情報を意味するものである。

HELO² クライアントのホストの識別名を送る(通常はホストの FQDN またはそれに準じる名前にする)

MAIL FROM 本来の差出人アドレスを指定する

RCPT TO 実際の受取人アドレスを指定する

これら 3 要素のうち HELO と MAIL FROM は発信者側に関する情報を受信側サーバに伝えるものであり、本来は正しい情報が伝えられるものである。しかし、SPAM 送信のような不道德行為をする場合は身元を明らかにしないかもしくは偽造するかのどちらかであることがほとんどである。したがって、「身元不明」である SMTP セッション、「身元偽造」と判断できる SMTP セッションを検知したときに、メッセージの受信を拒否するようにすれば受信側サーバに負荷を掛けることなく SPAM を撃退できるのではないかと予想できる。

²現在では拡張 SMTP 用の EHLO を送ることが多いが本稿では HELO/EHLO を代表する意味で HELO を利用する。

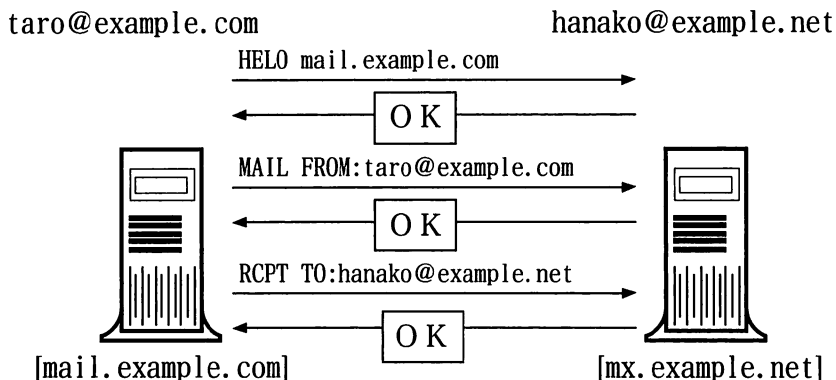


図 1: SMTP のセッション開始時

3 SMTP レベルでの拒否基準

送信元サーバの情報を元に、メッセージの本文を送り込まれる以前の段階で「受け取る価値がない」ものを予測し、受信を拒否する。そのために、送信してくるサーバのもつ

- ・送信サーバの IP アドレス
- ・SMTP セッションで送って来る HELO, MAIL FROM, RCPT TO

の情報に関して以下の判断基準を適用する。

1. 「IP アドレスのブラックリスト」に登録されたサーバからの接続か
2. HELO で指定された文字列が

(a) HELO 文字列ブラックリスト (badhelo) 中の特定のパターンにマッチするか

(“yahoo.com” , “hotmail.com” を HELO 文字列に指定するものは SPAM である)

- (b) ドットを含む FQDN らしき文字列になっているか
(“pc123” など個人PCから発せられるものは SPAM である可能性が高い)
 - (c) IP アドレスの場合、でたらめな IP アドレスになっているか
 - (d) 受信サーバの IP アドレスや FQDN、メイルドメインになっているか
(HELO で相手のアドレスを指定するのは不正なクライアント)
 - (e) 虚偽に使われやすくなおかつ交流の可能性の低い国別トップレベルドメインを名乗っているか
(このルールは厳しいので DNS 逆引きレコードが未定義の場合のみ適用する)
3. MAIL FROM が特定の送信者ブラックリスト (badmailfrom) に含まれるパターンにマッチするか
 4. MAIL FROM がドメイン部(@記号以降)を含む正しい形式か
 5. MAIL FROM のドメイン部が実在するドメイン名か
 6. RCPT TO が宛先ブラックリスト (badrcptto) に含まれるパターンにマッチするか

ここでいうブラックリストとは、SPAM を頻繁に送って来た実績のあるホストや送信者アドレス (群) などをあらかじめ登録したリストである。それとは逆に信頼できるホストを登録したものをホワイトリストという。

また、特殊なケースとして、利用者のうち誰かが以前利用していたメールアドレスを、現在利用しているアドレスに転送するような場合がある。このとき以前利用していたサーバで十分な SPAM 対策を施していない場合、数多くの SPAM が転送されて届くことになる。かといって、転送元の組織にあるメイルサーバを「IP アドレスのブラックリスト」に追加するわけにはいかない。そこで、典型的な転送元メイルサーバに関して、「転送を受理する送信者メールアドレス(のワイルドカード)」を指定できるようにした(ACCEPTDOMAINS 機能)。

実際に上記のルールを適用し、受信拒否を行なうメイルサーバを構築して実地試験を行なった。

3.1 SMTP anti-badmail の実装

qmail-1.03 [1] の SMTP デーモンプログラムである qmail-smtpd.c に 3 節の拒否ルールを処理できる実装を追加した。なお、各ルール実装のうち 5 は既存の mfccheck パッチ [4] を拡張し、「信頼できるクライアント」からの接続のときは全てのメールを受理するホワイトリスト機構を追加した。また、6 は既存の badrcptto パッチ [5] を拡張し、ブラックリストにドメイン部のみを指定すると、それに一致するもの全てを除外できるワイルドカード指定機能を追加した。

4 ブラックリストの作成

4.1 SPAM 送信の経験的動向分析

国内外を問わず SPAM 送信者は存在する。しかし、その SPAM 送信手法は国内外で大きく異なっていることが観察できた。

国外SPAM ダイヤルアップ・xDSL などの動的割当ホストから手当たり次第に送る。DNS 逆引き登録は未登録、またはプロバイダの動的割当ホスト名を指していることが多い。FROM アドレスは常に偽造する。全くランダムな FROM アドレスのこともあるが、ドメイン部を実在するものにしつつローカル部 (@記号より前) を乱数生成したものにすることが多い。HELO 文字列は乱数で生成したものや実在するドメインを詐称するものがある。

国内SPAM 国外 SPAM のような手段による SPAM はかなり少ない。これは、携帯電話に附随するメールの普及が日本で格段に進んでいるのが理由の一つと考えられ、SPAM 送信者のターゲットはより購読層の多い携帯電話系メールに移行しているためと考えられる。数少ない国内 SPAM で

は、正規に取得した無料プロバイダの正しいメールアドレスを利用して、正々堂々と送って来るものが散見される。これは送信手順が一般プロバイダの正規のメールサーバから送られるため、SMTP レベルでは悪意によるものかは見分けられない。

4.2 作成したブラックリスト

前節の SPAM 送信サーバの傾向をふまえ、SMTP サービスデーモンが参照するブラックリストを作成する。ブラックリストには以下の4種類が存在する。

badhelo 拒否したい HELO 文字列のリスト。yahoo.com など詐称されやすいもの、“pc1”, “server” など、初心者がインストール後放置したクライアント PC や、未熟な管理者が良く分からずに設定したサーバなどがもつ典型的なホスト名、通常はありえない受信サーバ側のホスト名・IP アドレス等を列挙する。部分文字列によるワイルドカード指定も可能。

badmailfrom 拒否したい FROM アドレスのリスト。@yahoo.com など詐称されやすいものや実際にしつこく送って来る FROM アドレスを列挙する。ワイルドカード指定も可能。ただし、@yahoo.com のような詐称されやすいものも、本当の Yahoo サーバから送られたものは信頼する必要があるので後述するホワイトリストに登録しておく。

badrcptto 拒否したい RCPT (宛先) のリスト。存在しない宛先にしつこく送って来る SPAM などを拒否するために指定する。qmail-1.03 が本来持つ機能である。ただし、自動的に送られる SPAM の中には、SPAM 自動送信プログラムなどによりでたらめな宛先アドレスを大量に生成して送りつけて来るものもある。このため、存在しないアドレスにメッセージが届いた場合は自動的に badrcptto データベースにそれを登録し、次回から SMTP セッションの段階で受信拒否できるようにする機構を用意した。

³実際には tcpserver のルールファイル

smtp.cdb SMTP 接続クライアントの IP アドレスまたはドメイン名に応じて「信頼すべき」か「受信拒否すべき」かを定義したリスト³。「信頼すべき」サーバとして、例えば yahoo.com の本物のサーバを登録すると、そこからは @yahoo.com という FROM アドレスを持つメールを受信許可できる。具体的にはデータベース中に

```
=.yahoo.com:allow,ACCEPTDOMAINS="@yahoo.com/.yahoo.com"
```

などを書くことで Yahoo のもつメールサーバからは @yahoo.com または @*.yahoo.com の送信アドレスをもつメッセージを全て受け取れることを指示できる。SMTP デーモンが起動しているときに環境変数 ACCEPTDOMAINS が定義されている場合、その値に/区切りで列挙されたドメインのみを受け取るようになっている

もう一点、利用者が別組織のメールアドレスを転送している場合、そこに送られて来る送信者ドメイン一覧を指定できる。たとえば、yuuji@ae.keio.ac.jp という宛先に届くメールは、ほとんどが .jp または .org ドメインのものであるため、@ae.keio.ac.jp のメールサーバから転送されて来るメールは、送信者が *.jp, *.org のものだけに限定するなどといったことができる。この場合データベース中には

```
=.ae.keio.ac.jp:allow,ADONLY="",ACCEPTDOMAINS=".jp/.org"
```

と書けば良い。環境変数 ADONLY が定義されている場合は、環境変数 ACCEPTDOMAINS に / 区切りで列挙されたドメインに該当する MAIL FROM のメッセージ以外は一切受け取らないようになる。

4.3 SMTP サーバに実装した拒否ルール

SMTP サービスデーモン (qmail-smtpd.c) に以下の拒否ルールを埋め込み実装した。

・FROM アドレスの検査

存在しないドメイン名、メールアドレスとして成立しないもの (@記号の無いもの⁴) は拒否する。

・HELO 文字列の検査

RFC 1123 では HELO に基づく受信拒否を禁止しているものの、明らかにでたらめと分かるパターンは SPAM だと判定できることは間違い無い。HELO でホスト名と見なせない、「ドットの無い文字列」、「ドットは含むが存在し得ないトップレベルドメイン」、「嘘の IP アドレス」などを HELO 文字列として送って来たクライアントからのメールは受信拒否する。

5 適用実験結果

筆者の運営する利用者数34名のサイトで2004年4月7日に接続を受けたSMTP 全ての受信許可/拒否結果は表1のようになった。

表1: 受信許可 / 拒否の実数

接続総数	受信許可	受信拒否
1574	1099(69.8%)	475(30.2%)

このうち受信許可したものの中には、実際には SPAM であるものが素通りしているものもある。これについて全利用者の協力を得て、受信した SPAM の報告を受けたところ、全7通の受信が確認できた。得られた報告が通過 SPAM の全てではない可能性はあるが、 $\frac{1099}{1099.7} \approx 0.993 = \text{約}99\%$ の SPAM が拒否できたことになる。

⁴ エラーメールに用いられる NULL アドレスなどは許可する

受信拒否となったものの拒否根拠の内訳は表2のようにになっている(複数根拠を含む)。

表2: 受信拒否根拠

MAILFROM による拒否		202
内訳	存在しないドメイン	25
	ブラックリスト (badmailfrom) にマッチ	202
RCPT TO による拒否		72
HELO による拒否		140
内訳	badhelo にマッチ	126
	ドットなし	13
	あり得ないドメイン	10
IP アドレスに基づく拒否		199

表3: 拒否した FROM アドレス上位 10

46	163.com
28	tom.com
21	hotmail.com
16	yahoo.com
11	sina.com
11	msn.com
5	tiscali.co.uk
5	ginko.de
4	sohu.com
4	21cn.net

MAILFROM による拒否では、著名な無料メールサービスのメイルドメイン名を詐称したものが目立つ。表3に、拒否根拠となったメイルドメイン名を示す。

逆に表1中の受信拒否したものには、本来なら受け取るべきものが含まれる可能性がある。これについて送信者アドレスと、実際の送信メールサーバの組み合わせを判断基準として、受け取るべきものだったかどうかの判定をしたと

ころ、交友があるとは思えない国からのもの、送信サーバと送信者ドメインが一致しないものといった組み合わせなどにより、中味を見なくとも SPAM と判定できるものばかりであった。よって、試験期日の受信拒否結果に関しては、受け取るべきものを拒否したものはなかったと判断できる。

6 検 討

かつて電子メールは、研究機関や企業などにいる限られた人のみを使うものであった。当時は便利なこの媒体を悪用する人間の比率が無視できる程度だったため、送られて来るメッセージは確実に自分に対して意味を為すものという前提がおけた。しかし現在では約半数が受け取っても意味のないものとなっている。全く知見のない国から発せられた SPAM が手元に届くことも多い。本論で試験を行なったサイトに関していえば、国内よりも海外発の SPAM の方が多い状況となっている。このような状況では、見ず知らずの国から SPAM が届いた場合、もしその国の人と親交を持つ人間が一人もいないような場合は、SPAM 発信元となったサービスプロバイダの持っているアドレスブロックを全て拒否対象としてしまっても実害はない。

このように、利用者の居住・活動地域を考慮してブラックリストを作ることには以下の得失が考えられる。

利点 SPAM が同じプロバイダネットワークから届く可能性がゼロになる。

欠点 同じプロバイダを使っている人から善意のメールが来た場合にも届かなくなる。

前者に関していえば、SPAM を自由に送ることができるようなネットワークでは、同様の悪用が再発する可能性が高いと予想できる。いっぽう、後者に関していえば、我々にとってなじみのない国の人間から交流を求めて来る可能性は皆無に等しい。

よって、地域性を考慮したブラックリストを独自に構築して、メールサーバの受信許可不許可の根拠とする方策は大きな正の効果をもたらすといえる。

7 結 論

SMTP 接続クライアントの DNS 登録状況、HELO/MAIL FROM/RCPT TO を調査することで数多くの SPAM を効果的に拒否できることが分かった。また、拒否根拠となるデータベースへの項目追加に際しては、交流の可能性の高低を考慮して登録するアドレスの範囲を決めることで、本来受け取るべき善意のメッセージを拒否してしまう可能性を下げることができた。

入手方法

今回使用した SMTP anti-badmail の実装は
<http://www.gentei.org/~yuuji/software/qmpatch/>より
入手できる。

謝 辞

ブラックリストの構築に共同参画し、効果的な拒否リストの構築に協力して下さった馬渡亮太氏に感謝致します。また、サーバを通り抜けて受信した SPAM を逐一報告して下さった gentei.org ドメイン全ユーザの皆様に感謝致します。

参考文献

- [1] D. J. Bernstein;
qmail; <http://cr.yip.to/qmail.html>
- [2] maps; <http://mail-abuse.org/>
- [3] SORBS; Fighting spam by finding and listing Exploitable Servers;
<http://www.dnsbl.au.sorbs.net/>
- [4] Nagy Balazs;
mfcheck-3; <http://www.qmail.org/qmail-1.03-mfcheck.3.patch>
- [5] Ward Vandewege;
badrcptto; <http://patch.be/qmail/badrcptto.html>
- [6] HIROSE, Yuuji;
qmail patches; <http://www.gentei.org/~yuuji/software/qmpatch/>
- [7] 広瀬雄二, 大駒誠一; SPAM 門前払い・SMTP レベルでの受信拒否方策の検証;
平成 15 年度第 2 回情報処理学会東北支部研究会資料番号 14

付 録

qmail-smtpd.c(qmail-1.03) に対する修正箇所を以下に示す。これは CVS の利用できる環境から、

```
cvs -d pserver:anonymous@yatex.org:/qmail \  
    rdiff -u -r qmail-1-03 -r anti-badmail \  
    qmail/qmail-smtpd.c
```

として得ることもできる。また、本論で実験を行なった時のブラックリストは

```
cvs -d :pserver:anonymous@yatex.org:/qmail \  
    co -D 2004/4/7 spamdb
```

により得ることができる。

qmail-smtpd.c からの差分 (unified diff)

```
Index: qmail/qmail-smtpd.c
diff -u qmail/qmail-smtpd.c:1.1.1.1 qmail/qmail-smtpd.c:1.1.1.1.2.22
--- qmail/qmail-smtpd.c:1.1.1.1 Sun Jan 12 16:57:58 2003
+++ qmail/qmail-smtpd.c Thu Feb 26 23:27:23 2004
@@ -23,11 +23,16 @@
#include "timeoutread.h"
#include "timeoutwrite.h"
#include "commands.h"
+#include "dns.h"
+#include <syslog.h>

#define MAXHOPS 100
unsigned int databytes = 0;
+unsigned int mfchk = 0;
int timeout = 1200;

+unsigned int paranoidchk = 0;
+
int safewrite(fd,buf,len) int fd; char *buf; int len;
{
    int r;
@@ -50,6 +55,9 @@
void straynewline() { out("451 See http://pobox.com/~djb/docs/smtp1f.html.\r\n"); flush(); _exit(1); }

void err_bmf() { out("553 sorry, your envelope sender is in my badmailfrom list (#5.7.1)\r\n"); }
+void err_brt() { out("553 sorry, your envelope recipient is in my badrcptto list (#5.7.1)\r\n"); }
+void err_hmf() { out("553 sorry, your envelope sender domain must exist (#5.7.1)\r\n"); }
+void err_smf() { out("451 DNS temporary failure (#4.3.0)\r\n"); }

void err_nogateway() { out("553 sorry, that domain isn't in my list of allowed rcpthosts (#5.7.1)\r\n"); }
void err_unimpl() { out("502 unimplemented (#5.5.1)\r\n"); }
void err_syntax() { out("555 syntax error (#5.5.4)\r\n"); }
@@ -58,7 +66,16 @@
void err_noop() { out("250 ok\r\n"); }
void err_vrfy() { out("252 send some mail, i'll try my best\r\n"); }
void err_qqt() { out("451 qqt failure (#4.3.0)\r\n"); }
-
+void err_rej(char *host) {
+ out("553 Sorry, I cannot talk with such a dubious mail server[";
+ out(host);
+ out("] (#5.7.1)\r\n"); }
+void err_ptr() { out("553 Your IP address's PTR record points to wrong hostname (#5.7.1)\r\n"); }
+void err_reqptr() { out("553 Sorry, we request correct PTR record in your address domain (#5.7.1)\r\n"); }
+void err_nofwd() {
+ out("553 I can accept only ");
+ out(env_get("ACCEPTDOMAINS"));
+ out("domains from your server (#5.7.1)\r\n"); }

stralloc greeting = {0};
@@ -96,6 +113,19 @@
```

```

int bmfok = 0;
stralloc bmf = {0};
struct constmap mapbmf;
+int brtok = 0;
+stralloc brt = {0};
+struct constmap mapbrt;
+
+int bhlok = 0;
+stralloc bhl = {0};
+struct constmap mapbhl;
+
+/* bad HELO check */
+int flagbahl = 0;
+
+/* ACCEPTDOMAINS result */
+int flagacceptdom = 0;

void setup()
{
@@ -112,11 +142,29 @@

    if (rcptheosts_init() == -1) die_control();

+ if (control_readint(&mfchk,"control/mfcheck") == -1) die_control();
+ x = env_get("MFCHECK");
+ if (x) { scan_ulong(x,&u); mfchk = u; }
+
+ if (control_readint(&paranoidchk,"control/paranoid") == -1) die_control();
+ x = env_get("PARANOIDCHECK");
+ if (x) { scan_ulong(x,&u); paranoidchk = u; }
+
bmfok = control_readfile(&bmf,"control/badmailfrom",0);
if (bmfok == -1) die_control();
if (bmfok)
    if (!constmap_init(&mapbmf,bmf.s,bmf.len,0)) die_nomem();
+
+ brtok = control_readfile(&brt,"control/badrcptto",0);
+ if (brtok == -1) die_control();
+ if (brtok)
+     if (!constmap_init(&mapbrt,brt.s,brt.len,0)) die_nomem();

+ bhlok = control_readfile(&bhl,"control/badhelo",0);
+ if (bhlok == -1) die_control();
+ if (bhlok)
+     if (!constmap_init(&mapbhl,bhl.s,bhl.len,0)) die_nomem();
+
if (control_readint(&databytes,"control/databytes") == -1) die_control();
x = env_get("DATABYTES");
if (x) { scan_ulong(x,&u); databytes = u; }
@@ -199,12 +247,125 @@

int bmfcheck()
{
+ /* Return 0: if it seems valid.
+ * Return 1: if it matches with an entry in control/badmailfrom

```

```

+ * Return 2: if doesn't have domain part (without @ mark)
+ */
+ int j;
+ char *ad;
+ if (!bmfok) return 0;
+ if (env_get("RELAYCLIENT") || env_get("RELIABLECLIENT"))
+     return 0; /* pass the reliable clients */
+ ad = env_get("ACCEPTDOMAINS");
+ /* $ACCEPTDOMAINS is a delimited list of acceptable mail domains.
+    Typical values are as follows;
+
+ @yahoo.com : accepts only @yahoo.com domain
+ .hotmail.com : accepts any @*.hotmail.com
+ @hotmail.com/.hotmail.com : accepts @hotmail.com and @*.hotmail.com
+
+ multiple domain list is delimited by '|'.
+
+ It would be good to put @hotmail.com and any major free-mail domain
+ list in /var/qmail/control/badmailfrom file while setting
+ ACCEPTDOMAINS variable in tcpserver's rule file like this;
+
+ =.hotmail.com:allow,ACCEPTDOMAINS="@hotmail.com"
+
+ With these settings, qmail-smtpd prevents any fake @hotmail.com.
+ The 'true' hotmail.com's mail servers can send @hotmail.com messages.
+ */
+ if (ad) {
+     char *ds=ad;
+     int len = str_len(ad);
+     int slash, at, dot, ok=0;
+     at = byte_rchr(addr.s,addr.len,'@');
+     while (ds < ad+len) {
+         slash = byte_chr(ds,ad+len-ds,'/');
+         /* compare with user@domain */
+         if (slash == addr.len-1)
+             if (!str_diffn(ds,addr.s,slash)) {
+                 ok = 1;
+                 break;
+             }
+         /* compare with @domain */
+         if (at < addr.len && slash == addr.len-at-1)
+             if (!str_diffn(ds,addr.s+at,slash)) {
+                 ok = 1;
+                 break;
+             }
+         /* compare with wild card matching */
+         for (dot=at; dot<addr.len; dot++) {
+             if (addr.s[dot] == '.')
+                 if (slash == addr.len-dot-1)
+                     if (!str_diffn(ds,addr.s+dot,slash)) {
+                         ok = 1;
+                         goto out;
+                     }
+         }
+     }
+     ds += slash+1;

```



```

+ }
+ out:
+ if (ok) {
+     flagacceptdom = 1;
+     return 0;
+ }
+ }
+ if (constmap(&mapbmf, addr.s, addr.len - 1)) return 1;
+ j = byte_rchr(addr.s, addr.len, '@');
+ if (addr.len > 1 && j >= addr.len)
+     return 2; /* when sender=<>, addr.len=1 */
+ if (j < addr.len)
+     if (constmap(&mapbmf, addr.s + j, addr.len - j - 1)) return 1;
+ /* .foo.bar (wild carding) */
+ for (; j < addr.len; ++j)
+     if (addr.s[j] == '.')
+         if (constmap(&mapbmf, addr.s + j, addr.len - j - 1)) return 1;
+ return 0;
+}
+
+int mfcheck()
+{
+    stralloc sa = {0};
+    ipalloc ia = {0};
+    unsigned int random;
+    int j;
+    int verp=0;
+
+    if (!mfchk) return 0;
+    if (env_get("RELAYCLIENT") || env_get("RELIABLECLIENT"))
+        return 0;
+    random = now() + (getpid() << 16);
+    if (str_equal(addr.s+addr.len-5, "##[)") /* Should we accept? */
+        return 0;
+    verp = str_equal(addr.s+addr.len-5, "-@[)");
+    if (verp) {
+        j = byte_chr(addr.s, addr.len, '@') + 1;
+    } else {
+        j = byte_rchr(addr.s, addr.len, '@') + 1;
+    }
+    if (j < addr.len) {
+        stralloc_copys(&sa, addr.s + j);
+        if (verp) { /* if sender is VERP seed */
+            sa.len -= 4;
+            sa.s[sa.len] = 0;
+        }
+        dns_init(0);
+        j = dns_mxip(&ia, &sa, random);
+        if (j < 0)
+            return j;
+    }
+    return 0;
+}
+
+int brtcheck()

```

```

+{
+ int j;
+ if (!brtok) return 0;
+ if (constmap(&mapbrt,addr.s,addr.len - 1)) return 1;
+ j = byte_rchr(addr.s,addr.len,'@');
+ if (j < addr.len)
+   if (constmap(&mapbrt,addr.s + j,addr.len - j - 1)) return 1;
+   return 0;
+ }

@@ -216,9 +377,63 @@
+   return r;
+ }

+int helocheck(arg) char *arg;
+{
+ /* Return 0: if it seems valid,
+  * Return 1: if it matches badhelo
+  * Return 2: if not dots in helo string
+  * Return 3: invalid tld
+  * Return 4: matches with badhelo:unknown
+  */
+ int j;
+ if (env_get("RELAYCLIENT") || env_get("RELIABLECLIENT"))
+ return 0; /* Relay Client's are allowed to use bogus HELO host */
+
+ if (!bhlok) return 0;
+ if (constmap(&mapbhl,arg,str_len(arg))) return 1;
+ j = str_chr(arg, '.');
+ /* no dots, reject it */
+ if ('\0' == arg[j])
+ return 2;
+ if (arg[j])
+   if (constmap(&mapbhl,arg + j,str_len(arg) - j)) return 1;
+ /* .foo.bar (wild carding) */
+ for (;arg[j];++j)
+   if (arg[j] == '.')
+     if (constmap(&mapbhl,arg + j,str_len(arg) - j)) return 1;
+ /* random helo with dots */
+ j = str_rchr(arg, '.');
+ if ((str_len(arg)-j) > 5)
+ return 3;
+ if (arg[0] >= '0' && arg[0] <= '9' /* in case it begins with digit */
+ && arg[j+1] >= '0' && arg[j+1] <= '9') { /* && last segment begins with digit */
+ struct ip_address ip; /* bogus IP address check */
+   if (ip_scan(arg, &ip) { /* arg seems to be decimal IP-address */
+     if (str_diff(arg, remoteip))
+       return 1; /* IP in HELO differs */
+   }
+ } else { /* in case it begin wiht non-digit */
+   if (arg[str_len(arg)-1] >= '0' && arg[str_len(arg)-1] <= '9')
+     return 3; /* last char of tld can't be a digit */
+ }
+ /* unknown helo check */
+ if (str_equal("unknown",remotehost)) {

```

```

+ static stralloc unkohelo = {0};
+ if (!stralloc_copys(&unkohelo,arg) die_nomem());
+ if (!stralloc_cats(&unkohelo,":unknown") die_nomem());
+ if (constmap(&mapbhl,unkohelo.s,unkohelo.len) return 4;
+ /* wild card matching */
+ for (j=str_chr(unkohelo.s,'.');unkohelo.s[j];++j)
+   if (unkohelo.s[j] == '.')
+     if (constmap(&mapbhl,unkohelo.s+j,unkohelo.len-j)) return 4;
+
+ }
+ return 0;
+}

int seenmail = 0;
int flagbarf; /* defined if seenmail */
+int flagmfcheck;

stralloc mailfrom = {0};
stralloc rcptto = {0};

@@ -226,11 +441,13 @@
{
  smtp_greet("250 "); out("\r\n");
  seenmail = 0; dohelo(arg);
+ flagbahl = helocheck();
}

void smtp_ehlo(arg) char *arg;
{
  smtp_greet("250-"); out("\r\n250-PIPELINING\r\n250 8BITIME\r\n");
  seenmail = 0; dohelo(arg);
+ flagbahl = helocheck();
}

void smtp_rset()
{
@@ -241,6 +458,7 @@
{
  if (!addrparse(arg)) { err_syntax(); return; }
  flagbarf = bmfcheck();
+ flagmfcheck = mfcheck();
  seenmail = 1;
  if (!stralloc_copys(&rcptto,"") die_nomem());
  if (!stralloc_copys(&mailfrom,addr.s) die_nomem());
@@ -248,22 +466,169 @@
  out("250 ok\r\n");
}

void smtp_rcpt(arg) char *arg; {
+ int flagbadhost = 0;
+ int rely=0;
+ int somethingbad = 0;
+ static stralloc badreason = {0};
+ if (!stralloc_copys(&badreason,"") die_nomem());
  if (!seenmail) { err_wantmail(); return; }
  if (!addrparse(arg)) { err_syntax(); return; }
- if (flagbarf) { err_bmf(); return; }
+ if (flagmfcheck) {

```

```

+ somethingbad = flagmfcheck;
+ switch (flagmfcheck) {
+ case DNS_HARD:
+   err_hmf();
+   stralloc_copys(&badreason,"DOMAIN");
+   break;
+ case DNS_SOFT: err_smf(); return;
+ case DNS_MEM: die_nomem();
+ }
+
+ if (flagbarf) {
+   char *types[] = {"", "_noDOMAIN"};
+   char *type = types[(flagbarf-1)%((sizeof types)/(sizeof (char*)))];
+   if (somethingbad) {
+     if (!stralloc_cats(&badreason,"/") die_nomem();
+   } else {
+     /* We produce SMTP ERROR only once. Ditto in latter err_XXX() */
+     err_bmf();
+   }
+   if (!stralloc_cats(&badreason,"MAILFROM") die_nomem();
+   if (!stralloc_cats(&badreason,type) die_nomem();
+   somethingbad = flagbarf;
+ }
+
+ rely = (flagacceptdom||env_get("RELAYCLIENT")||env_get("RELIABLECLIENT"));
+ flagbarf = brtcheck();
+ if (!rely && flagbarf) {
+   if (somethingbad) {
+     if (!stralloc_cats(&badreason,"/") die_nomem();
+   } else {
+     err_brt();
+   }
+   if (!stralloc_cats(&badreason,"RCPT") die_nomem();
+   somethingbad = flagbarf;
+ }
+
+   if (relayclient) {
+     --addr.len;
+     if (!stralloc_cats(&addr,relayclient) die_nomem();
+     if (!stralloc_0(&addr)) die_nomem();
+   }
+   else
-   if (!addrallowed()) { err_nogateway(); return; }
+   if (!addrallowed()) {
+     if (somethingbad) {
+       if (!stralloc_cats(&badreason,"/") die_nomem();
+     } else {
+       err_nogateway();
+     }
+     if (!stralloc_cats(&badreason,"RELAY") die_nomem();
+     somethingbad = 1;
+   }
+
+   /* Reject all but in $ACCEPTDOMAINS */
+   if (!rely && (int)env_get("ADONLY") {
+     if (somethingbad) {
+       if (!stralloc_cats(&badreason,"/") die_nomem();
+     } else {

```

```

+     err_nofwd();
+     }
+     if (!stralloc_cats(&badreason,"ADONLY") die_nomem());
+     somethingbad = 1;
+     }
+     /* badhost check */
+     flagbadhost = (int)env_get("BADHOST");
+     if (!rely && (flagbahl || flagbadhost)) {
+     char *types[] = {"HELO", "HELO_noDot", "HELO_random", "HELO_unknownbad"};
+     char *type;
+     if (flagbadhost) {
+         type = "HOST";
+     } else {
+         type = types[(flagbahl-1)%((sizeof types)/(sizeof (char*)))]};
+     }
+     if (somethingbad) {
+         if (!stralloc_cats(&badreason,"/") die_nomem());
+     } else
+         err_rej(helohost.s);
+     if (!stralloc_cats(&badreason,type) die_nomem());
+     somethingbad = 1;
+     }
+     if (!rely && env_get("REQPTR") && str_equal("unknown", remotehost)) {
+     if (somethingbad) {
+         if (!stralloc_cats(&badreason,"/") die_nomem());
+     } else
+         err_reqptr();
+     if (!stralloc_cats(&badreason,"REQPTR") die_nomem());
+     somethingbad = 1;
+     }
+     if (!rely && paranoidchk) {
+     if (env_get("TCPPARANOID") { /* This requires `tcpserver -p' */
+         if (somethingbad) {
+             if (!stralloc_cats(&badreason,"/") die_nomem());
+         } else {
+             err_ptr();
+         }
+         if (!stralloc_cats(&badreason,"PTR") die_nomem());
+         somethingbad = 1;
+         openlog("qmail-smtpd", LOG_PID, LOG_MAIL);
+         syslog(LOG_INFO, "Paranoid: helo=[%s], host=%s[%s], sender=[%s], rcpt=[%s]", helohost.s,
+ env_get("TCPPARANOID"),remoteip, mailfrom.s, addr.s);
+         closelog();
+     }
+     }
+     if (!stralloc_0(&badreason) die_nomem());
+ #if 1
+     if (!str_equal(remoteip, "127.0.0.1")) {
+ #include <sys/stat.h>
+ #include <fcntl.h>
+     struct stat st;
+     static stralloc log = {0};
+     char *logpipe = "control/qmlog";
+     openlog("qmail-smtpd", LOG_PID, LOG_LOCAL1);
+     if (-1 != stat(logpipe, &st)

```

```

+  && st.st_mode&S_IWUSR
+  && st.st_mode&S_IFIFO) {
+  if (!fork()) {
+  int pipe = open(logpipe, O_WRONLY|O_NDELAY);
+  if (-1 != pipe) {
+  if (!stralloc_copys(&log,"QLV_01: ") die_nomem();
+  if (somethingbad) {
+  if (!stralloc_cats(&log,"Rejected_") die_nomem();
+  if (!stralloc_cats(&log,badreason.s) die_nomem();
+  } else {
+  if (!stralloc_cats(&log,"Accepted") die_nomem();
+  }
+  if (!stralloc_cats(&log,": ") die_nomem();
+  if (!stralloc_cats(&log,"helo=") die_nomem();
+  if (!stralloc_cats(&log,helohost.s) die_nomem();
+  if (!stralloc_cats(&log," remotehost=") die_nomem();
+  if (!stralloc_cats(&log,remotehost) die_nomem();
+  if (!stralloc_cats(&log," remoteip=") die_nomem();
+  if (!stralloc_cats(&log,remoteip) die_nomem();
+  if (!stralloc_cats(&log," sender=") die_nomem();
+  if (!stralloc_cats(&log,mailfrom.s) die_nomem();
+  if (!stralloc_cats(&log," rcpt=") die_nomem();
+  if (!stralloc_cats(&log,addr.s) die_nomem();
+  if (!stralloc_cats(&log,"\n") die_nomem();
+  if (!stralloc_0(&log) die_nomem();
+  write(pipe,log.s,log.len); /* Don't care its failure */
+  close(pipe);
+  }
+  exit(0);
+  }
+  }
+  syslog(LOG_INFO, "%s: helo=%s, host=%s, remoteip=%s, sender=%s, rcpt=%s, relayclient=%s",
somethingbad ? "Rejected": "Accepted", helohost.s, remotehost, remoteip, mailfrom.s, addr.s,
(env_get("RELAYCLIENT") ? "yes" : "no"));
+  closelog();
+  }
+endif
+  if (somethingbad) {
+  openlog("qmail-smtpd", 0, LOG_MAIL);
+  syslog(LOG_INFO, "bad %s: host=%s, remoteip=%s, helo=%s, sender=%s, rcpt=%s, badreason.s,
remotehost, remoteip,helohost.s, mailfrom.s, addr.s);
+  closelog();
+  seenmail = 0;
+  return;
+  }
+  if (!stralloc_cats(&rcptto,"T") die_nomem();
+  if (!stralloc_cats(&rcptto,addr.s) die_nomem();
+  if (!stralloc_0(&rcptto) die_nomem();
+  out("250 ok\r\n");
+  }
-
int saferead(fd,buf,len) int fd; char *buf; int len;
{

```