第 XXXI 部

迷惑メール低減に関する技術開発と普及

第 31 部
迷惑メール低減に関する技術開発と普及

第 1 章　はじめに

Antispam ワーキンググループは、迷惑メールを低減させる技術の発展と普及を目的として 2005 年に設立された。本年度の主な活動は以下のとおりである。

- ドメイン認証に関する普及率の測定
- SPF を普及させるための提案
- OP25B を普及させるための提案

第 2 章　ドメイン認証に関する普及率の測定

ドメイン認証の普及率を客観的に示すために、送信側の普及率を測定している。これは、JPRS の共同研究であり、JPRS から .jp 以下のドメインリストを提供して頂き、各ドメインに認証情報が宣言されているか検査する。対象としているドメイン認証は、SPF、Sender ID、DomainKeys、そして、DKIM である。この測定は、2005 年 4 月から開始し、毎月実施している。

表 2.1. ドメイン認証の普及率に関する調査結果

年月	SPF											DomainKeys										
	AD	AC	CO	GO	OR	NE	GR	ED	地域型	汎用	合計	AD	AC	CO	GO	OR	NE	GR	ED	地域型	汎用	合計
2006/01	17	89	10181	22	725	336	377	70	46	2777	14642	1(3)	0(1)	6(14)	0(1)	2(6)	2(5)	2(4)	0	0(2)	5(54)	18(90)
2005/12	17	87	10191	22	721	333	375	70	45	2691	14553	1(3)	0(1)	5(11)	0	1(4)	2(5)	2(4)	0	0(2)	4(50)	15(80)
2005/11	13	23	673	2	97	88	24	8	15	841	1785	1(3)	0	4(12)	0	1(4)	2(5)	2(3)	0	0(1)	2(44)	12(72)
2005/10	13	22	634	2	87	83	21	8	14	771	1656	1(2)	0(1)	2(6)	0	1(2)	2(5)	0(2)	0	0(1)	2(37)	8(56)
2005/09	11	20	595	2	77	86	20	5	11	751	1579	1	0	2	0	1	1	0	0	0	1	6
2005/08	10	11	268	1	21	77	15	0	8	540	951	1	0	3	0	1	1	0	0	0	1	7
2005/07	10	10	236	1	20	68	14	0	8	489	856	1	0	3	0	1	1	0	0	0	1	7
2005/06	8	10	219	1	19	67	14	0	6	435	779	1	0	3	0	1	1	0	0	0	1	7
2005/05	6	10	204	0	18	66	13	0	8	389	714	1	0	3	0	1	1	0	0	0	1	7
2005/04	4	8	180	0	16	59	12	0	4	310	593	1	0	1	0	1	1	0	0	0	1	5

※ 2005 年 4 月〜6 月までの DomainKeys は集計方法に誤りがあったため、7 月の値に基づき補正してあります。

年月	登録数											MX										
	AD	AC	CO	GO	OR	NE	GR	ED	地域型	汎用	合計	AD	AC	CO	GO	OR	NE	GR	ED	地域型	汎用	合計
2006/01	297	3251	283873	839	21071	17317	8948	4394	3816	439784	786124	250	3061	263452	720	19686	13183	7602	3946	3235	292504	608649
2005/12	297	3237	282451	841	20930	17324	8983	4379	3819	432446	777239	253	3057	262943	724	19611	13218	7634	3938	3258	288708	604341
2005/11	297	3231	280872	836	20830	17307	9008	4381	3846	426275	769445	253	3046	261402	723	19487	13216	7665	3933	3291	274666	588671
2005/10	297	3217	279529	834	20723	17285	9043	4371	3880	419485	761287	254	3034	259130	727	19389	13188	7687	3962	3153	265489	577012
2005/09	298	3210	278159	832	20625	17330	9086	4359	3921	413768	754213	252	3032	258982	726	19310	13218	7707	3951	3248	266774	578190
2005/08	298	3203	276621	835	20487	17316	9122	4348	3960	406542	745381	257	3022	257207	723	19154	13177	7740	3944	3425	264961	574586
2005/07	299	3192	275219	839	20352	17299	9149	4327	4010	397161	734520	257	3003	255199	722	18971	13178	7762	3916	3445	255087	562522
2005/06	299	3184	273574	837	20222	17321	9177	4316	4061	388552	724233	258	2997	254282	727	18892	13180	7782	3901	3360	238204	544568
2005/05	299	3166	272223	835	20088	17318	9203	4315	4084	361104	719858	255	2986	252860	726	18758	13222	7832	3894	3381	234123	539053
2005/04	301	3161	270323	841	19921	17308	9207	4315	4151	352856	685168	256	2967	250595	726	18557	13203	7838	3881	3487	224528	527035

2.2 測定方法

ドメイン認証の導入は、送信側と受信側に大別できる。この内、あるサイトが受信側としてドメイン認証を導入しているか、外部から知る方法はない。一方、送信側はDNSで認証情報を公開するので、認証情報の有無を調べることにより、送信側の導入を判断できる。

WIDEプロジェクトは、JPRSと共同研究契約を結び、2005年4月からドメイン認証の普及率を毎月測定している。JPRSからは、.jp以下のドメインの一覧を提供して頂き、それらのドメインに対して認証情報の有無を検査している。

たとえば、JPRSから提供されたドメイン名の1つがexample.jpだとする。このサイトが、メール用のドメインとしてexample.jpを使っているのか、サブドメインを設定しているのかはわからない。しかしながら、最近では大学などを除くとサブドメインを設けないのが一般的である。事実、JPRSから提供されたドメインを検査したところ、約8割にMXリソースレコード（以下RR）が設定されていた。すなわち、全体の約8割は、サブドメインを設けずにメール用のドメインとして利用されている可能性が高い。このため、サブドメインを推測せずに、JPRSから提供されたドメイン名のまま、ドメイン認証の普及率を測定しても、全体の傾向を掴むには十分であると考えられる。

調査対象とするドメイン認証は以下のとおりである。

- SPF、Sender ID
- DomainKeys、DKIM

調査は、JPRSから提供して頂いたマシン上で実施している。リゾルバとして、ローカルのbindを用い、bindに再帰処理をさせて各ドメインの権威サーバからデータを得ている。

2.2.1 SPF、Sender ID

SPFもSender IDも、ドメイン名に対してSPF RR（type 99）を宣言する。ただし、現時点ではSPF RRが普及していないため、TXT RRで代用するのが一般的である。以下に、SPFとSender IDの宣言の特徴を示す。

- SPFは、SPF RRのデータとして“v=spf1”を持つ
- Sender IDは、SPF RRのデータとして“spf2”を持つ（v=がないことに注意）

Sender IDの受信側では、“v=spf1”というSPF RRを“spf2.0/mfrom,pra”として取り扱うことが決まっている。そのため、SPFとSender IDを厳密に分けることにはあまり意味がない。そこで、我々の調査では両者を同一視している。

測定方法は、ドメイン・リストの各要素に対し以下を繰り返す。

- 「ドメイン名」に対し、SPF RRを引く
  - SPF RRが応答セクション（answer section）に含まれていれば、導入している
  - そうでなければ、導入していない
- 「ドメイン名」に対し、TXT RRを引く
  - TXT RRが応答セクションに含まれており、かつそのデータが文字列“spf”を含んでいれば、導入している
  - それ以外なら、導入していない

2.2.2 DomainKeys、DKIM

DomainKeysの送信側では、公開鍵をDNSで公開する。たとえば、example.jpに対する公開鍵の名前は、<selector>._domainkey.example.jpとなる。<selector>の部分は、受け取ったメールの署名を見なければわからない。<selector>は推測できないので、公開鍵の有無からDomainKeysの導入を判断することはできない。

古い測定方法

DomainKeysでは、ポリシーもDNSで宣言できる。このポリシーの名前は、_domainkey.example.jpであり、推測する部分はない。そこで、2005年4月の調査開始時点では、ポリシーの有無で導入を調べることにした。ただし、ポリシーの宣言はオプションであるため、ポリシーを宣言せずにDomainKeysを導入しているサイトは落としてしまう。

新しい測定方法

2005年10月からは、測定方法を改良し、サブドメイン“_domainkey”が存在するか否かで、導入を判断することにした。この方法では、ポリシーを宣言せずにDomainKeysを導入しているサイトも拾える可能性がある。

ところで、サブドメインの設定には、以下の方法

がある。

1. _domainkey.example.jpという個別のゾーンを設定する
2. example.jp ゾーンに_domainkey.example.jpを書く

_domainkey.example.jp という名前の SOA RR が存在すれば、1. であると判断できる。一方、_domainkey.example.jp の SOA RR を引いて “name error” となれば、1. でも 2. でもないと判断できる。

それ以外の場合、_domainkey.example.jpの TXT RR を引いて、“no error” となれば、2. である可能性がある。ただし、実際には _domainkey.example.jp が存在しないのにも関わらず、*.example.jp に何らかの RR が定義されているために “no error” となった可能性もある。そこで次に、*.example.jp の TXT RR を引く。これが “name error” となれば、*.example.jpは定義されていないので、_domainkey.example.jp が存在しているとわかる。

_domainkey.example.jp と *.example.jp が共に “no error” となった場合は、以下の可能性がある。

- _domainkey.example.jp が存在し、*.example.jp も存在する。
- _domainkey.example.jp が存在せず、*.example.jp が存在する。

これらを区別するためには、これら 2 つの名前に対し、いくつかの RR（たとえば TXT RR）を引いてみて、それぞれに対し両者の返答内容が完全に一致するか判定すればよい。なんらかの不一致が見つかれば前者である。完全に一致するなら、後者の可能性が高い。

新しい測定方法のまとめ

以上をまとめ、新しい測定方法では、ドメイン・リストの各要素に対し以下を繰り返す。

- 「_domainkey.ドメイン名」の SOA RR を引く
  - “no error” で、SOA RR が応答セクションに含まれていれば、導入している。次のドメイン候補へ。
  - “no error” で、応答セクションが空であれば、以下へ。
  - “name error” なら、次のドメイン候補へ。
- 「_domainkey.ドメイン名」の TXT RR を引く
  - “no error” なら、応答セクション（a と呼ぶ）を保存して、以下へ。
- 「*.ドメイン名」の TXT RR を引く
  - “name error” なら、導入している。次のドメイン候補へ。
  - “no error” なら、応答セクション（b と呼ぶ）を保存して、以下へ。
- (a) と (b) を比較し、一致すれば導入してない。一致していなければ導入している。

現実問題として、おかしな振る舞いをする DNS サーバも存在するため、実際の測定方法はもう少し複雑である。また、古い測定方法との継続を考慮して、新しい測定方法でもポリシーの有無も調べている。

2.2.3 DKIM

DKIMでは、ポリシーの名前が_policy._domainkey.example.jp であること以外は、DomainKeys と同じである。サブドメイン _domainkey の存在の有無では、DomainKeys と DKIM を区別することはできない。よって、我々の調査では両者を区別していない。

第 3 章　SPF を普及させるための提案

3.1 要旨

本報告書では、SPF を普及させるために以下のように提案する。

- “~”（softfail）の意味を「受け取らなければならないが、エラーメールは返さなくてもよい」と定義する。
- DNS で SPF リソースレコード（以下「RR」）を宣言する際は、all のプレフィックスとして（“-” の利用がためらわれる場合）“~” を選択する。“?” は使用すべきではない。
- user unknown などの理由によりエラーメールを返す際に、送信アドレスに対する SPF の検査が softfail または fail である場合は、エラーメールを返さない。

3.2 SPF の普及に関する問題点

SPF は、メールアドレスのドメイン部分を認証するドメイン認証の一種である。その安全性は、DNS

のそれに依存する。

SPF では、あるドメインに対する送信サーバの IP アドレスを宣言する。受信側は、SMTP MAIL FROM に指定された送信アドレスからドメインを切り出し、DNS を検索して、そのドメインに対する送信サーバの IP アドレスを得る。これを、SMTP コネクションの相手側の IP アドレスと比較することで、ドメイン部分の正当性を検証する。

このように、SPF のしくみは、送信側の宣言と受信側での検証からなり、サイト（ドメイン）では独立して導入できる。おおざっぱな言い方ではあるが、SPF では、送信サーバを宣言するドメインが増えなければ、受信側で検証できるメールの比率も増加しない。SPF の普及のためには、まず送信サーバを宣言するドメインを増やしていく必要があるが、それには以下の問題点があると考えられる。

- 送信サーバを宣言しても、普及率が低い時期はメリットが少ない。逆に、宣言の設定を誤るとメールが受け取ってもらえなくなるリスクを伴う。これらを比較し、宣言しない方を選択してしまいがちである。
- 送信サーバを宣言するとしても、all のプレフィックスとして “?” を選択する。これは、all に合致した場合、宣言してないことと同じであるため、受信側のメリットを損ねている。

そこで本報告書では、送信サーバを宣言すると、自分のドメインに届く不要なエラーメールが少なくなる方法を提示する。現在、エラーメールのほとんどはドメインを詐称されることによって発生しており、これらの不要なエラーメールは受信サーバに大きな負荷をかけている。この負荷が軽減されることは、大きなメリットとなるはずである。このメリットにより、送信サーバを宣言するドメインの数が増加することを期待する。

3.3 予備知識

本報告書での提案を理解しやすくするために、まずハーベスティングについて説明する。

ハーベスティングとは、メールアドレスを収集する行為をいう。まず、Web などに公開されているメールアドレスを収集する方法が挙げられる。また、あるドメインの受信サーバにメールを送るふりをして、実際にユーザが存在するか調べる方法もある。そのメールアドレスのリストは、乱数／総当たり的に生成されたもの、人名などの辞書を使うもの、そして Web から収集したものなどが考えられる。

受信サーバに問い合わせるハーベスティングは、以下のような手順が一般的である。まず、SMTP MAIL FROM に詐称したメールアドレスを指定する。次に、SMTP RCPT TO に対しメールアドレスのリストから 1 つを選んで指定する。メールサーバから「要求が正常に終了した」という応答（250）が返れば、そのメールアドレスが有効であると判断する。この後、可能な限り SMTP RCPT TO を繰り返し、リストを検証していく。

ハーベスティングを防ぐために、受信サーバのいくつかは SMTP RCPT TO で指定されたメールアドレスのユーザ名が有効であろうとなかろうと、常に 250 を返しメールを受け取るよう設定されている。この種の受信サーバは、エラーメールを自分自身で生成し送信する。ハーベスティングに対するエラーメールは、多くの場合 SMTP MAIL FROM に詐称したメールアドレスが指定されているため、返すこと自体が詐称されたドメインにとって迷惑である。事実、ハーベスティングによって引き起こされる大量のエラーメールは、受信サーバに大きな負荷をかける DoS となっている。

また多量の不要なメールは、ウイルスによっても引き起こされる。ウイルスが送るメールは、送信者が詐称されていることが多い。このメールの受信者が存在しない場合は、エラーメールが詐称された送信者へ返る。また、受信側でウイルスの検知を通知するタイプのアンチウイルス製品を利用している場合、検知メールが詐称された送信者へ送られる。

3.4 提案

SPF をエラーメールの低減に役立てるために、以下のように提案する。

まず、all のプレフィックスである “~”（softfail）であるが、仕様では「neutral と fail の中間として扱う」と書かれているだけであり、意味が曖昧である。そこで、softfail の意味を「受け取らなければならない（受信は拒否してはならない）」とする。また、ハーベスティング対策を施した受信サーバでは、「エラーメールは返さなくてもよい」と定義することを提案する。

ハーベスティング対策を施した受信サーバの動作を以下のように変更するよう提案する。受信側では、

まず SPF の検証結果を保存しておく。次に、user unknown などの理由でエラーメールを返す際に、保存しておいた SPF の検証結果を参照する。これが softfail あるいは fail であれば、送信アドレスは詐称されているのだからエラーメールを返さないようにする。これにより、送信サーバを宣言していれば、ドメインを送信アドレスに悪用されても、それに対するエラーメールは届かなくなる。

SPF RR を使い送信サーバを宣言する際は、all のプレフィックスとして “~” か “-” を利用する。最初は “~” が妥当であろう。“?” と宣言しているドメインは、なるべく早く “~” に変更する。

3.5 考察

エラーメールのほとんどはハーベスティングやウイルスによって引き起こされており、ほとんどが無意味である。これを低減させることには抵抗が少ないはずである。

エラーメールが減るというメリットがあれば、SPF RR を宣言するサイトが増えるだろう。受信側で SPF を活用しエラーメールを返さないサイトが増えれば、ますますエラーメールは減っていく。このような好循環から、送信側も受信側も SPF の導入が進むことを期待したい。

SPF は、SMTP レベルの転送と相性が悪い。メールが転送されて届いた場合、SPF の検証はドメインが詐称されているという結果となる。しかしながら、本報告書の提案した “~” が宣言されていれば、転送されたメール自体は受信される。転送先で user unknown となった場合は、本来返るべきエラーメールが返らないが、そもそもその転送の設定は間違っていると考えられ、設定自体を直すべきである。

第 4 章　OP25B を普及させるための提案

4.1 概要

迷惑メールの問題を解決するために、メールリーダがメールを送信する際は、現在利用されている SMTP ポート（25 番）ではなく、投稿ポート（587 番）を利用することが望まれている。ここでは、メールリーダの送信時の動作を「最初に投稿ポートの利用を試み、接続できない場合は SMTP ポートへ接続する」よう変更することを提案する。また、その変更はユーザから見た操作性にはほとんど影響を与えないことを示す。

4.2 背景

迷惑メール配送業者は、Bot（ゾンビ）を使ったり、規制の甘い ISP を渡り歩いたりして、PC から目標とするメールサーバに直接 SMTP コネクションを張り、メールを送信する。これを繰り返すことによって、大量の迷惑メールをばらまく。

この対策として、境界で上り方向の SMTP ポートを遮断する ISP が増えてきた。この対策は、OP25B（Outbound Port 25 Blocking）と呼ばれることもある。

OP25B 対策を実施した ISP は、もちろん、その ISP の送信サーバから他の ISP の受信サーバへ、25 番ポートを使い SMTP コネクションを確立できるように設定する。このような環境では、ISP のユーザは、ISP の送信サーバを使う限り、メールの送信に問題は生じない。

しかしながら、他の ISP にも正式にアカウントを持ち、他の ISP の送信サーバを利用してメールを送信するユーザもいる。この通信は、迷惑メール配送業者のと同様に遮断される。このユーザを救うために、投稿と配送の分離が提案されている。投稿とはメールリーダから送信サーバへメールを配送することであり、配送とは送信サーバから受信サーバへメールを届けることである。

投稿も配送もプロトコルとしては、これまでどおり SMTP を使う。投稿は、OP25B に遮断されないように、投稿（Submission; RFC2476[93]）ポートである 587 番を使う。一方、配送はこれまでどおり SMTP ポートである 25 番を使う。

ポート番号を 25 から 587 へ変えただけでは、迷惑メール配送業者もそれを使うようになるだろう。そこで、投稿ポートでは必ずユーザ認証（SMTP AUTH）を要求しなければならない。また、配送に対しドメイン認証の技術が普及すれば、メールアドレスを詐称できない環境が整うことになる（図 4.1 を参照のこと）。詳しくは、メールアドレスの詐称を防止する「ドメイン認証」[333] を参照されたい。

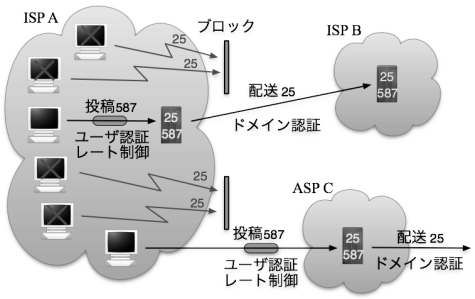

図 4.1. メールアドレスを詐称できない環境

4.3 投稿ポートとユーザ認証

現在、投稿ポートを提供する ISP は増加している。一方で、デフォルトで投稿ポートを利用するメールリーダは皆無である。

4.4 課題

現在、多くのメールリーダが、メールを送信する際に使うポートを変更可能である。しかし、ユーザが自分で変更しなければならない現状では、投稿ポートの普及速度は遅いであろう。投稿ポートを早く普及させるためには、メールリーダがデフォルトで投稿ポートを利用するようになることが望まれている。

しかしながら、デフォルトのポートを単純に投稿ポートに設定してリリースすると、送信サーバが投稿ポートでメールを受け付けてない場合、メールの送信に失敗するという問題が起こる。

4.5 フォールバックとその技術考察

上記の課題を解決するために、まず投稿ポートの利用を試み、利用できない場合は SMTP ポートへフォールバックするという方法を提案する。

以下、場合を分けて動作を説明する。

1) 投稿ポートが提供されている場合

何も問題がない。OP25B の環境では、断然有利である。

2) TCP RST が返る場合

RTT の時間だけ待たされて、SMTP ポートへフォールバックする。ユーザにはほとんど分からない。多くの ISP が、このケースであると思われる。

3) ICMP ハードエラーが返る場合

RTT の時間だけ待たされて、SMTP ポートへフォールバックする。ユーザにはほとんど分からない。ファイアウォールでフィルタリングされている場合に起こりうる。

ちなみに、ハードエラーは以下のとおり：

- ICMPv4 Destination Unreachable
  - コード 2 = protocol unreachable
  - コード 3 = port unreachable
  - コード 4 = fragmentation needed and DF set
  - コード 6 = destination network unknown
  - コード 7 = destination host unknown
  - コード 8 = source host isolated
  - コード 9 = communication with destination network administratively prohibited
  - コード 10 = communication with destination host administratively prohibited
  - コード 11 = network unreachable for type of service
  - コード 12 = host unreachable for type of service
- ICMPv6 Destination Unreachable
  - コード 1 = communication with destination administratively prohibited
  - コード 4 = port unreachable

4) ICMP ソフトエラーが返る場合

ICMP ソフトエラーは、経路制御の問題から引き起こされる。経路制御の問題が起こっているときは、通信が不可能である。投稿ポートを先に試すか否かにかかわらず、通信が不可能であることに変わりはない。また、実際問題として、ほとんど起こらないと思われる。

以下に、ICMP ソフトエラーを示す。

- ICMPv4 Destination Unreachable
  - コード 0 = net unreachable
  - コード 1 = host unreachable
  - コード 5 = source route failed
- ICMPv6 Destination Unreachable
  - コード 0 = no route to destination
  - コード 3 = address unreachable

提案した方式にはあまり意味がないかもしれないが、念のため TCP が ICMP ソフトエラーを受けた際にどう振る舞うか場合を 2 つに分けて説明する。

4.1) TCP の古い実装

コネクション確立前も、ソフトエラーをソ

フトエラーとして扱う。TCP の再送に上限があれば、それに制約される。BSD の場合は、10 秒程度である。上限がなければ、TCP がタイムアウトする間待たされる。

4.2) TCP の新しい実装

コネクション確立前は、ソフトエラーをハードエラーとして扱う。すなわち、RTT の時間だけ待たされて、SMTP ポートへフォールバックする。ユーザにはほとんどわからない。Linux などがこの方式を採用している。

5) TCP がタイムアウトする場合

TCP タイムアウトの時間待たされて、SMTP ポートへフォールバックする。ファイアウォールが投稿ポートへのパケットを黙って破棄する場合に起こると思われる。

各 OS での TCP タイムアウトにかかる時間を表 4.1 に示す。

この場合においても、(non-blocking connect() を利用し）バックグラウンドでメールを送る場合は、ユーザにとってメールの配送が少し遅いと感じる程度であろう。Solaris のように TCP タイムアウトが長い場合は、メールリーダが短いタイマーを設定してもよい。

またメールリーダは、タイムアウトが起こった事実をネガティブキャッシュして、次から同じ環境で同じサーバへメールを送信する場合は、最初から SMTP ポートへ接続するようにしてもよい。あるいは、タイムアウトが起こった時点で、ダイアログを表示し、ユーザの指示を仰いでもいいだろう。

蛇足になるが、この内容は最近普及し始めている greet pause とはなんら関係ない。greet pause は SMTP レベルの話であり、この小節は TCP レベルの話である。(greet pause は、サーバが accept() の後に数十秒待って write() するのだから、クライアントの connect() は block されず、read() が block される。)

表 4.1.　TCP タイムアウトにかかる時間

OS	TCP タイムアウトにかかる時間
BSD	75 秒
Linux	189 秒
Solaris	225 秒
Windows XP	21 秒

4.6 実装とフィールド実験

すでに、Mew[180] でバックグランドでのフォールバック機能を実装し終え、考察どおりに動作することを確認した。また、各ケースの割合を把握するために、フィールド実験 [181] を実施した。

実験で集まったレポートは 39 である。その内、localhost または「自宅のサーバ」と書かれていたサーバの数が 7。これは一般的なユーザのケースでないとして排除すると、総数は 32。これには、ユーザとサーバの組が同じで、クライアントの位置だけ違うレポートも含まれる。それらを吟味した結果、すべて同一のケースと見なせることがわかり、ユーザとサーバの組で冗長性を排除すると、総数は 26 となった。

統計的に十分な調査ではないが、ある程度の傾向はわかると考えられる。

総数 26 の内、サーバの内訳を表 4.2 に示す。

ケースの内訳を表 4.3 に示す。

この実験を通じて得た知見：

- ケース 5 は、無視できない割合で存在する。上記の実験では約 15%。そのためメールリーダは、OS の長い TCP タイムアウトを回避するために、短いタイマーを設定すべきである。
- 投稿ポートは予想外に普及している。上記の実験では約 46%。しかも、ユーザ認証なしで提供されている場合が多いため、迷惑メール配送業者に悪用される可能性がある。

表 4.2.　サーバの内訳

ISP	16
ASP	1
会社	6
大学	2
不明	1

表 4.3.　ケースの内訳

ケース 1（投稿ポート）	12
ケース 2（TCP RST）	7
ケース 3（ハードエラー）	0
ケース 4（ソフトエラー）	0
ケース 5（TCP タイムアウト）	4
SSH（22）	2
SSL（465）	1

- メールリーダは、サーバの返す AUTH ケーパビリティに従ってユーザ認証を実行する必要がある。投稿ポートではユーザ認証を要求され、フォールバック後、SMTP ポートでは要求されない場合にも、対応できるように実装する必要がある。
- ただし、ユーザ認証に対応していないにもかかわらず、AUTH ケーパビリティを返すサーバも存在する。よって、ユーザ認証を無効化するオプションも必要である。
- ポートレベルのフォールバックだけでなく、IPv6 から IPv4 へのフォールバックも必要である。すなわち、投稿ポート/IPv6 →投稿ポート/IPv4 → SMTP ポート/IPv6 → SMTP ポート/IPv4 とフォールバックするよう実装すべきである。

4.7 結論

フォールバック技術を用いれば、投稿ポートをデフォルトにしても、ユーザに大きな影響を与えることはない。

バックグラウンドでメールを送信できるメールリーダの場合、ほとんどユーザには影響を与えないだろう。

フォアグラウンドでしかメールを送信できないメールリーダの場合、利用環境によってはフォールバックに時間がかかる。ただし、その時間はメールリーダが短く設定可能である。次からは SMTP ポートのみを利用する方法も考えられる。

4.8 備考

フォールバックのアイディアは、IPv6 と IPv4 のデュアルスタック環境では、一般的である。フォールバックに対する考察は、長氏による解説[1]が詳しい。

付録 A　Mew の実装方法

Emacs 21 までは、blocking connect() しか利用できなかった。blocking connect() では、C レベルで IPv6 から IPv4 へフォールバックする。Emacs 22 からは、non-blocking connect() も利用できる。ただし、non-blocking connect() では、実装上の制約により IPv6 から IPv4 へのフォールバックに対応できていない。

そこで、non-blocking connect() を利用する際は、Emacs Lisp レベルでアドレスファミリを指定し、C レベルで利用されるアドレスファミリを明示的に限定する方法を採用した。

現時点での Mew の実装は、以下のとおり。

- IPv4 を指定し投稿ポートへ non-blocking connect()
  - タイマーは 10 秒に設定
- 上記が失敗した場合、SMTP ポートへ blocking connect()
  - C レベルで IPv6 から IPv4 へフォールバック

最初に「IPv6 を指定し投稿ポートへ non-blocking connect()」を加えると対称的になるが、現状ではタイムアウトの時間が増えるだけの環境が多いので、採用していない。

上記の手続きは、TLS や SSH トンネルとも共存できるように実装してある。SSL を利用する際は、465 ポートへ blocking connect() を用い、SMTP ポートへはフォールバックしない。（SMTP over SSL のためのポート番号は、正式には割り当てられていない。465 ポートは不用意に用いられ、実質的な標準となっている。465 ポートは、正式には Cisco のあるプロトコルに割り当てられていることに注意。）

付録 B　Windows での実装

Windows には、セキュリティソフトウェアが通信を監視する仕組みがあるので、メールリーダの SMTP ポートへの通信を投稿ポートへ、セキュリティソフトウェアがリレーする方法が現実的かもしれない。この場合、セキュリティソフトウェアがフォールバックを実装する必要がある。

1　http://www.v6fix.net/docs/v6fix.html.ja#sec4

第 5 章　おわりに

普及率の測定により、2006 年 12 月に SPF（Sender ID）の普及が大幅に進んだことが判明した。このように普及率を把握することは、有意義であるといえる。

SPF と OP25B を普及させるために、それぞれで新たな技術を提案した。特に SPF に対する提案は重要であり、SPF の普及を加速すると期待できる。

このように antispam WG は、本年度、十分な成果を得たといえる。