長野大学紀要　第21巻第1号　47－62頁　1999

ネットワーク・セキュリティの現状と課題（Ⅰ）
——メール配送方式の問題と対策（SPAM問題を中心に）——

Aspects of Network Security (Part I)
—— Mail System (focused on the SPAM problem) ——

平　岡　信　之*
Nobuyuki　HIRAOKA

［概　要］

インターネットを中心とした情報通信システムのセキュリティについて、主としてシステム管理者の視点から、シリーズで論じていく。第1回ではSPAM問題をとりあげ、その対策について論じる。また、SPAM対策の一環として筆者が試作した小さなプログラムについて詳細に報告する。

1　背景

1.1　昨今のインターネット

インターネットはこの数年の間に、その規模が飛躍的に増大したのに加えて、形態においても大きく変化した。その変化は以下のように要約できる。

a アプリケーションの多様化：WWW、インターネット電話やビデオ会議等のリアルタイム双方向通信、或はラジオ・ビデオのような放送、といった多様なアプリケーションが出現した。現在のキラーアプリケーションはWWWであると言われているが、今後の変遷は（WWWがそうであったのと同様に）予測困難である。ただし、その変化の中でも、電子メールは、娯楽や息抜きでなく重要な連絡手段と位置づける人が多く、（量的なトラフィック占有率は低下しているものの）依然として重要な位置を占めている。

b ユーザおよび目的の多様化：ユーザ層が、特殊な立場の層から一般市民に広がった。その利用目的も、研究をはじめとするいわば公益的（個々のユーザにとっては私益的な面は多々あったにしても）な目的から、ビジネスや趣味などの、純粋に私益的な目的での利用へと広がってきた。

一方、上記の様々な変化に係わらず、変化しないものも勿論ある。それを1つあげるとすれば、主に定額負担の形をとるコスト負担モデルであろう。エンドユーザがプロバイダにダイヤルアップする場合の通信料金、或は10円メール[1]等の特殊なサービス、といった限定した部分を除けば、従量的課金は行われない、というのがインターネット利用コスト負担の典型的な形態であり、これによりユーザが大らかな心持ちで利用できてきたことが、これまでのインターネットの大きな普及要因の1つであることは想像に難くない。事実、現在も例えば電子メールの送信にあたって、1通あたり或は1オクテットあたりといった従量課金は行われていない。本稿で扱う問題は、こうした状況を背景として発生したものである。

1.2　本論文の経緯と目的

筆者は8年前に本学に赴任した時から、当時は教員によるボランタリな活動として行われていた学内ネットワークの構築管理の仕事に参加している。[2][3]その後、ネットワーク管理の仕事は幾つかの紆余曲折を経て、現在はその基幹業務は専門の学内組織に委ねられるようになったが、筆者は学内その他にテストベッド的環境を維持し、その

*助教授　nobu@nagano.ac.jp

構築管理を通じてネットワークにおける技術的課題等を研究してきている。本論文はその過程で得た知見をまとめたものと位置づけることができる。

本論文では、SPAM問題の状況や筆者の対策を報告することにより、システムの管理運用に携わる方々への何らかの参考になることを目的とし、また同時に、昨今の技術のあり方に関する議論の題材を提供することもそのねらいとする。

なお、本論文では、Unixオペレーティングシステムおよび TCP/IP プロトコルに関する若干の知識を持つ読者を想定していることをご承知頂きたい。

1.3 現在のメール配送の仕組み

議論の前提として、メールの配送が一般にどのように行われているかを、ここで示しておく。以下の図では3種類のソフトウェアが使われる。MUA (Mail User Agent；メール読み書きのためにユーザが直接使うソフトウェア) はユーザ機 (一般に PC) 上でクライアントとして動作し、MTA (Mail Transfer Agent；発信者主導で転送を担当) 及びMDA (Mail Delivery Agent；受信者主導で配送を担当) はプロバイダ等で管理されるサーバ機上で動作する。このMTAとMDAの管理が電子メールに関するシステム管理者の重要な業務である。また、スプール (Mail spool 或は Mail drop 等とも呼ばれる) は、ユーザ別に割り当てられたサーバ機上のディスク領域である。図で二重矢印は同一機上でのファイル I/O を示し、一重矢印はネットワーク上での通信を示す。投函と転送に広く使われるアプリケーション層プロトコルは SMTP[4] であり、配送には POP[5] が多く使われている (最近は IMAP[6] の利用も進みつつある)。

```
発信者のMUA－>MTA－> … －>MTA＝>スプール
          投函    転送      (書込み)
受信者のMUA<－MDA<＝スプール
          配送    (読出し)
```

2 SPAMについて

2.1 SPAMとは何か

無差別に大量の受信者に向けて発信されるメッセージをインターネット関係者は SPAM[7] と呼んでいる。以下のような形で見かけられる。

a ネットニュースにおいて、数多くのニュースグループに向けて同一内容の投稿を行う。

b 大量の受信者アドレスに向けて同一内容の電子メールの送信を行う。特にこれをUCE、或はUBE 等と呼ぶ場合もある。

c チャットのような場所で個人的メッセージをやはり多数の受信者に向けて送る。

相互コミュニケーションを行うための開放的なコミュニティがあれば、それを悪用する者が発生するようである。上記の中で、特にbの電子メールの世界におけるSPAMは、後述のようにシステム管理者にとっても厄介な課題を提供するものであり、また電子メールを重要なライフラインとするユーザが多いため話が深刻にする。本稿では、この、電子メールを用いたSPAMに話を限定する。なお、SPAMを発信する者をSPAMmerと呼ぶこともある。SPAMの語源等についての説明[8]は、ここでは省略する。

SPAMを発信する目的は様々だろうが、実利のあるなしの区別により大別すると

・自己顕示欲によるものや愉快犯など

・公告ビジネスおよびそれに類するものとして (ネズミ講への勧誘等も含まれる)

の2種類が推察できる。これがビジネスとして成立する背景として、前述のように、メールを発信することに対して従量的な課金が行われないという、インターネットのコスト負担モデルがあることは勿論である。SPAMとして散布されるメールの内容は様々だが、筆者の観察によれば、アダルトサイトへの誘いといったものが目立つようである。

散布対象となる受信者名簿は、ヤミ経路で不正に入手する以外に、大きなプロバイダのユーザ名生成のパターンを分析して、小さなプログラムにより自動生成する、といった方法も可能であろう。この場合、不送達メールは高い率で発生するだろうが、それはSPAMmerにとって大きな問題

にはならない。まさに「下手な鉄砲も…」がSPAMのビジネスモデルであり、発信コストが限りなくゼロに近いため、99%の受信者が眉を顰めようとも、1%の応答者がいればビジネスが立派に成立するだろうと推察できる。

2.2 SPAMによる脅威と迷惑

SPAMは、TCP/IPのプロトコル階層で言えば、最も上位にあたるアプリケーション層の機能の悪用である。現在のインターネットは、少なくともその基幹部分を形成するインフラストラクチャにおいては、非常に大容量のトラフィックを処理しうるものが稼働されており、たかだか数人とか数十人といったSPAMmerの動作で大きなダメージを受けることはない。

一方、末端サイトにおいては、以下のような被害が発生する。

- SPAMを受信する個々のユーザにとって、そのメールを処理するための時間（人的資源）や記憶領域などの物理的資源を浪費される。これは個人単位では些細な量であるが、ネットワーク全体でトータルすると、大量の資源浪費であると考えられる。
- 関係するサイトにとっては、システム管理者に宛てて、エラー報告や苦情のメールが殺到することになり、これらを処理するために人的資源の大幅な浪費を強いられる。また、こうして増殖したメールトラフィック等により、当該システムの性能が大幅に低下したり、何らかのシステムダウンを引き起こしたり、という結果を招くこともある。

要約すると、SPAMは、

- 当該システムに対する脅威
- 全体的に見た資源浪費

の2点において、犯罪的な行為であると言える。

また、実質的に脅威とならなくても、この行為は、インターネット全体で共有されるべき資源を、個人の利益のための不当に利用し、また浪費する行為であり、これは窃盗と同等の行為であると考えて、倫理的な観点からこれを忌避し糾弾する人は多い。

ただし、SPAM送信行為が、実際に法律に照らして犯罪を構成するかどうかは、判断の難しいところのようであり、また、これは各国の法律によっても違うだろう。1名の見知らぬ他人にメールを送ること自体は犯罪にはなりえない。それが100名であった場合、或は100万名であった場合、何通を越えれば犯罪とされるべきなのか、或はそれをどういう基準で判断すべきなのか、こういった法学的な問題がついてまわることになるだろう。さらに、これが法律的にみて犯罪を構成するものであったとしても、インターネットのボーダレスな性格を考えると、それを検挙することは非常に困難であろう。従って当面は、ユーザによる自衛と、インターネットコミュニティ全体での技術的対策で対応していくしかないだろう。

2.3 SPAMmerのバリエーションと対策

SPAMの受信を拒否し、或はそれを撲滅することを望む立場の者（ここでは「対策者」と呼ぶことにする）と、SPAMmerの手口との間には、ウイルスとワクチンの関係に似た、ある種のイタチゴッコ的関係が成立しつつあり、SPAMmerの手口は、以下の3項に示すように、より複雑に変容しつつある。これはセキュリティ問題全般に当てはまることだが、対策者は、攻撃者の現在の手口を理解した上で対策をたてる必要がある。

2.3.1 単純な無差別散布

プロバイダ等から交付された自分のアドレスを使って、単純にメールの散布を行う、即ち、発信者アドレスに関して細工を施さない、いわばシンプルなSPAMである。これに対して対策者は、SPAMと思われるメールを自動的に破棄するよう、ユーザ単位、或はサイト単位で設定することが可能である。以下のような幾つかの対策が考えられる。

a SMTPによるコネクション発生時に、接続要求元のIPアドレスを検知し、SPAM発信元と思われるサイトからのSMTP接続或は転送要求を拒否する（システムレベルでの対策）。

b 転送要求を受理したメールに関して、そのヘッダから発信元アドレスを検知し、SPAM発信元と思われるサイトからのメールを破棄する（システムレベルでの対策）。

c 上記bをユーザレベルで行う。

2.3.2 踏み台の利用

SPAMの増加に対応して、MTAも進化する。最近のMTAでは、設定ファイル（いわばブラックリストである）を参照して、そのファイルにリストされた特定のアドレスを持つホストからのメール転送を拒否するようになっている。こうした情報を収集しリアルタイムで参照できるようにしたサイト（RBLと呼ばれる）[9]もあり、また、これを活用するための機能を組み込む拡張が行われたＭＴＡも存在する[10]。こういった機能がSPAMを食い止める柵として働く。それに対して、SPAMmerは、その柵をかいくぐるための次の策を生み出した。踏み台となるホストの利用である。

一般にあるサイトの玄関口にあるマシンにおいて、MTAを通過するメールの流れは、下記の4種類に分類できる。

ａサイト内からサイト内へ
ｂサイト内からサイト外へ
ｃサイト外からサイト内へ
ｄサイト外からサイト外へ

このうちａ～ｃは、サイト内のユーザが受信または発信（またはその両方）に関与するものであり、このMTAが責任を持って中継する必要のあるものである。が、ｄについては、このMTAが中継しなければいけない義理はない。かつてのバケツリレー型ネットワークの時代には、自サイトに関係ないメールを中継してあげることも必要があったが、現在はこの外部サービスは原則として必要なものではない。この機能を利用して、例えばmyself%my.dom.ain@some.do.mainといったアドレスを使って、外部のユーザがメールの反射テストを行うことができるというような便宜を外部に提供することができるが、これ自体、本質的なものではない。

インターネットは自律分散的管理の行われる、多様なネットワークである。その中には、良心的ユーザで構成されるインターネット、を前提とした牧歌的な時代の感覚で管理されているサイトが多々残っている。この主のサイトのMTAは上記のｄの流れかたをするメールもチェックなしで中継（リレー）する。SPAMmerは、このようなサイトのマシン（で、他のサイトでブラックリスト扱いされていないもの）を踏み台として利用する訳である。この場合、メールは［発信者のPC］－＞［踏み台］－＞［受信者サイトのMTA］といった経路で、上記の柵をかいくぐって送られる。踏み台にされたホストは、場合によっては他サイトのブラックリストに載せられ、自サイトから発信された良心的ユーザのメールを受理してもらえなくなる可能性がある。これを防ぐため、SPAM時代のサイト管理者は、MTAが上記ｄの流れのメールの中継を原則として拒否するよう設定しなければならない。

2.3.3 アドレス詐称

2.3.1で対策ｂとして述べた柵をかいくぐるために、メールのヘッダに書かれる発信者アドレスとして、ニセのアドレスが書かれたSPAMも見かけられるようになってきている。全節と同様に、管理のゆるいMTAはこういうメールも中継してしまう。ニセアドレスとしては

ａ架空のドメイン
ｂ実在するドメインの架空のユーザ
ｃ実在するユーザ

がありうる。ｂやｃにおいて、アドレスを勝手に使われたユーザやサイト管理者は、ある日突然、大量の苦情メールやエラーメールを受け取ることになり、厄介なことにこのSPAMは（踏み台にされる場合とは違って）自サイトのMTAを通過しないので、検知したり防御したりする手段がない。大量に届くメールを（その中に隠れている必要なメールを失わないように注意しながら）フィルター等により自動破棄して、システムダウンといった重大な事態に陥ることを防ぐ、という防衛策と並行して、

・管理のゆるいMTAの管理者に苦情を述べる
・私（我が組織）は加害者でなく被害者であることを言明する

といった受動的な対策しかできない。

ビジネスの一環でメールを用いる場合、相手からの応答が得られる必要がある。特にSPAMを（依頼者として）用いるようなビジネスにおいては、それはいわば「飛込みの営業」の代りをなすものであり、相手からの応答がなければ話にならない。電子メールしかない時代ならば、そのため

にメールのヘッダに自分のアドレスを発信元として明記しておくことは不可欠（実際にはReply-toヘッダの利用といった裏技はあるにせよ）だったかも知れないが、今、実際には必ずしもそれは必要ではない。本文の中にURLを記述して、「このWebページを参照されたし」と書けばよい訳である。実際、多くのSPAMはこういった書き方がされている。

2.3.4　組織内からの発信

本項は、SPAMの変容過程ではなく、可能性として想定すべき別のケースとして、付記するものである。自組織内からSPAMを発信する者がある場合は、前項と同様の脅威が発生する。さらに組織の信用の失墜というような無形の（しかし甚大な）被害も発生し得る。また、中継システムのレベルでの検知や対策が困難であることも、前項と同様である。

しかし、だからといってユーザの通信を、内容に立ち入ってまで監視する訳にはいかず（検閲に相当する可能性があるため）、そこでシステム管理者のできる仕事としては、

1 メール転送ログの監視
2 素性の知れたMTA以外からの、SMTPによる外部へのコネクションの発生を、検知またはフィルターにより除去する

といったことであろう。また、この問題に対しては、

・ユーザ教育
・利用規定の整備

といった（技術的でない）対策が重要となるだろう。

2.4　メールシステムに対するその他の脅威

本稿ではSPAMに話を絞ることにしたが、実際にはSPAMによる被害はメールシステムにまつわる脅威の1つでしかない。本稿の主題とははずれるが、メールのセキュリティを考える上で考慮すべきその他の事項を、本節に簡単にまとめておくことにする。なお、これらは、システム（或はシステム管理者）にとっての脅威と、一般ユーザにとっての脅威とに分類することができる。ただし実際にはこれらはその両分類の間で相互に波及するものであり、システムが受ける被害はユーザに及ぶし、その逆もまた然りである。

a クラッキング（システムへの攻撃）：システムの脆弱な部分（セキュリティホール）をついて、外部から資源の利用権、特にその根本となるスーパユーザ権限を不正に取得し、またそれを利用する行為である。メール転送系にまつわるセキュリティホールは残念ながらこれまで殆ど毎年のように発見されている。

b 盗聴、改竄、成りすまし（ユーザへの攻撃）：この3つはセキュリティや暗号系の教科書に殆どいつもセットで現れる事項である。詳しい説明は省略する。その対策としてメッセージの暗号化が有力であり、公開鍵暗号系の応用が進みつつある[11]。

c メール爆弾（ユーザへの攻撃）：メールに添付した文書を開くと、その文書に仕込まれた自動化機能（マクロなど）が働き始め、その機能によりユーザのシステム破壊等の攻撃を行うというもの。怪しい文書は開かなければよい訳だが、セキュリティに関する意識の低いユーザは、無頓着にこれを開いてしまい餌食にされる。

d 誹謗中傷、いやがらせ（ユーザへの攻撃）：不愉快な（例えば猥褻な）表現を含むメールを送り付けたり、プライベートな事項（場合によってはメールアドレスも含む）を他人が勝手に公開する、等、様々なケースがある。パソコン通信など、発信者が特定できる（ある程度管理された）ネットワークにおいては、名誉毀損等にあたるものとして法的措置がとられてきた実績がある。インターネットにおいては前述のように発信元アドレスを詐称できてしまう場合もあり、発信者を特定して法的手段に訴えるといった対策は必ずしも可能であるという保証はないようである。

e 量による攻撃（両方）：SPAMも結果的には量による攻撃の一種であるが、その目的は広く散布することにある。その一方で、特定のホスト或は特定のユーザ宛てに大量のメールを集中的に送る、という形の攻撃もある。行為者にとっては、前述のSPAMによるダメージと同様ものを、より効率的に受信者に与え

ることができる。特に、マシンの性能やネットワーク接続の帯域といった能力面で、送信側の環境が受信側のそれよりも勝っている時には、受信側の受けるダメージは大きい。この行為の動機としては、怨恨や愉快犯が考えられる。

f 内発的な脅威（様々）：ここまでは、外部からの何らかの能動的な行為者がもたらすリスクであるが、それ以外に、攻撃に起因しない脅威として、システムダウン、ディスクのクラッシュ、管理者の操作ミス、等、様々な要因がメールシステムに対する脅威として考えられる。さらに、天変地異などの大きな事態までセキュリティ対策において考慮にいれるかどうか、はそのコストの大きさと絡んで難しい問題である。システム管理者とは、大変な仕事である。

3 対策の検討

3.1 問題のありか

前述の状況を考慮すると、MTAを管理する者にとっては、とにかくチェックを厳しくして、怪しいメールは中継しないようにする、というのが、自サイトの安全のためにも、また他のサイトに対する責任という点でも、妥当（かつ必要）な方針だということになる。全章で述べた状況を考えると、2.3.2項のdのようにサイトの外部から外部へ宛てたメールの転送を拒否し、かつ、発信アドレスが自サイトのもの以外のメールを外部に送らないようにする、といった設定をしなければならない。

しかし一方で、インターネットの利用形態は別の方向でも多様化してきている。PDAの進化やノートPCの軽量化によりモバイルコンピューティングが広がってきており、旅先からのメール読書き、或は職場のアドレスでのメール読書きを自宅から行うユーザ等、様々なアクセス形態が生まれつつある。管理者から見ると、自サイトにアドレスを持つユーザは今や世界中の至る所からメールを発信するという状況である。これらモバイルユーザ（ここでは自宅からのアクセス者等も含めて考える）は、どのMTAにメールを投函するのが正しいのだろうか。上記のSPAM対策がきっちり行われたとすると、

- モバイルユーザが接続しているアクセスプロバイダのMTAは、見知らぬアドレスを発信元とするメールの投函を拒否するだろう。
- 自組織のMTAは、組織外からのどこかから接続されたSMTPコネクションでの外部向けリレー要求を拒否するだろう。

現在の電子メールに係わるプロトコル体系では、安全と便宜とが両立しえないようである。

3.2 長期的解決

本節では、SPAM問題および上記の問題を根本的に解決し、或はその発生を非常に困難にするための抜本的対策を検討する。本節に示す対策は、方式の変更など、インターネットコミュニティ全体で検討実施する必要のある、単独のサイト管理者の努力では実施不可能なレベルの事項である。ここにすべての可能性を尽くしてあるとは言い難いので、今後さらに検討と議論は必要であろう。

3.2.1 信頼できるゲートウェイ通過印の埋め込み

現在のメール転送ソフトウェアは、中継するメールに、通過点に関する（原則として通過点あたり1行の）情報を挿入することになっている。この情報は、メールがフェイル（宛先に到達できないこと）した際に、発信者やシステム管理者が宛先アドレスやシステム設定のチェックをするために使われている。現在これらは平文で記述されている。この情報を各ゲートウェイを通過したという印（以下、仮に「消印」情報と呼ぶ）として中継ソフトウェアが認識し、以下のようなメールが転送されてきた時には自動的にそれを破棄するような機能を実現することができる。

a 転送要求のあった（送信されてきた）メールの中で、相手サイトの正しい消印情報の含まれていないもの。

b エラー通知や苦情等で帰ってきたメールの中で、自サイトの正しい消印情報の含まれていないもの

これによりSPAMの脅威を大幅に軽減できるだろう。

このためには、この消印は偽造不可能なもので

ある必要がある。上記bの目的に限定するなら秘密鍵暗号系が、a目的にも用いるならば公開鍵暗号系が原理的には利用可能である。aの目的でこれを適用するにあたっては、性能と暗号の強度を考慮した適切な桁数の設定など、現実的検討をした上で、消印の方式を規格化する必要があるだろう。一方、bの目的に限定する場合は、サイト単独で実施は可能である。

3.2.2 プロトコル体系の見直し

電子メールが使われ始めた当初は、電子メールのユーザとは、MTAの動作するマシン上にログインアカウントを持つユーザのことであり、メールの投函とは、同一マシン上でMTAソフトをプログラムとして起動することに他ならなかった。その転送過程は、1.3節の図式に対して、

発信者のMUA=>MTA−> … −>MTA=>スプール
（呼び出しによる投函）

と表現できる。このとき最初のMTAへの要求は、ログイン時の認証を終えたユーザに限定されていることが保証されているため、ユーザのチェックは行われない。また、MTAからMTAへの転送時にも、システム管理者同士の信頼関係のようなものがベースにあったため、厳しいチェックは行われていない。こういう背景で生まれたSMTPは、個々のユーザを認証する仕組みを持ち合わせていない。後に、認証のための枠組みを提供するための拡張は提案されているが、その決定的な実装が具体的に現れなかったためか、広く利用はされていない。

後になって、PCが普及し、MTAと離れたところ（別のマシン、即ちユーザのPC）上でMUAを動作させるために、MDAプロトコルとしてPOPが、そしてMDAソフトウェアのPOPサーバが使用されるようになっている。POP自体はユーザ認証機構を持つが、その目的メール受信者によるメールの「読み」作業に限定されている。PC上のMUAからのメールの投函には、元来MTA相互間のプロトコルであったSMTPが流用されて今に至っている。この、ユーザ認証を行わない投函プロトコルが、前節のように問題を複雑にしている要因の1つであるため、何らかのユーザ認証付きの投函プロトコルを策定してその実装を普及させていけばいい。そのプロトコルの候補としては、以下のいくつかの方向のものが考えられる。

- aSMTPを拡張する
- bMDAプロトコルを投函にも用いるために拡張する
- c新しいプロトコルを設計する
- d別のプロトコルを借用する（例えばHTTP[12]など）

なお、本稿を執筆中に、本節で扱ったのと同様の内容をサベイした上で上記のうちのaとして（しかしある意味でcに近いとも言えるが）MUAから最初のMTAへのMessage Submission（本稿では投函という訳語を用いている）に関する方式の提案がRFC[13]として出されていることがわかった。この提案でMUAが最初にメールを投函するサーバをMSA（Message Submission Agent）として、従来のSMTPを拡張した形の認証を行MUAとMSAの間で行うべきことを要求している。これにそった形でMUAとMSAの実装が出そろえば、インターネット全体として、この方向に向かって行く可能性が高いことが予想される。この場合の転送過程は、

発信者のMUA−>MSA−> … −>MTA=>スプール
投函（MSAプロトコル）

といった図式のものになるだろう。

3.2.3 配送しないメール

現在のインターネットは、その末端部分（すなわちダイヤルアップユーザとアクセスポイントの間）を除いてほぼ完全に常時接続が実現されている。SMTPに基づく電子メール配送が開始された頃はそうではなかったため、必然的に、電子メールは、そのコンテンツをエンベロープ情報とともに受信者のところ（スプールホスト）に送り届けることを前提にその方式が設計・実装され、現在もなおそのモデルに基づいて、電子メールが運用されている。つまり物理的郵便物と同じモデルが用いられている訳だが、現在のインターネットにおいては、情報伝達はこのモデルに限定される必

要はない（このことはWWWの普及という事実により実証されている）。最小限のエンベロープ情報（メールが送信されたという通知など）だけを受信者の（メールスプールに代る）所定の場所に届け、コンテンツは情報発信者のところにとどめておいても構わない訳である。この方法は、

a バケツリレー式の（現行の）メール配送系に欠けている機能の代表として、受信者がスプールの内容にアクセスしたかどうか（メールを読んだかどうか）を発信者が確認できないことがあげられるが、この方式では、その機能が容易に実現できる。

b メール発信における資源負担を、受信者よりも発信者に、より多く課する構造であり、昨今のSPAMビジネスに対して抑制効果を持つ。

といった利点を持つことが期待できる。

送信（この用語も考え直す必要があるが）されるコンテンツの置き場所は、外部からアクセス可能な場所である。従って、想定される受信者以外の者がここにおかれたコンテンツにアクセスすることを防ぐ何らかの仕組み必要であるが、これは、例えば以下のような手順を踏むことにより実現可能であろう。

1 発信者は本文を適当な鍵で（秘密鍵暗号系で）暗号化し、自分の発信スプールに置く。

2 発信者はその秘密鍵を受信者の公開鍵で暗号化してエンベロープ情報に含めて相手の受信スプールに届ける。

3 受信者は自分の（公開鍵暗号系の）秘密鍵により本文の鍵を複合化し、それを用いて発信者の手元にある本文を複合化する。

これはすでにPGPなどで実用化されている技術の、使い場所だけを変更するだけのものであり、即ち、要素技術としてはすでに出そろっていると考えられる。

3.3 短期的解決

前節で論じた解決策は、メール配送のしくみのレベルでの対策であり、前述のように、インターネット全体でコンセンサスを得て進めて行く必要のあるものである。それに対して、本節では、3.1節のような問題に対して、システム管理者が、サイト単位で実施することのできる対策について考えることにする。

3.3.1 VPN

自サイトから（ネットワーク的に）離れた場所にあるPCやLANから、自サイト内に向けて送られる（またはその逆の）パケットを、自サイトの管轄外のネット（プロバイダやプロバイダ間インタチェンジ）を通過する間だけ、さらに別のIPパケット内に（多くの場合、暗号化した上で）カプセル化して送る技術（あるいはそのように構成されたネットワーク）をVPN（Virtual Private Network）と呼ぶ。モバイルユーザと自サイトの間をVPNで接続することにより、モバイルユーザのマシンが（IPアドレスの点で）あたかも自サイト内にいるように見える。これにより3.1節にあるような問題は回避できる。

VPNは、ネットワーク層（IP層）を、上位層（仮想的IPネットワーク）と下位層（物理的ネットワーク）を積み上げて構成したものとみることができる。ユーザ認証は上位ネットワーク層で行われる。メール投函におけるユーザ認証をアプリケーション層で行う代りに、それを上位ネットワーク層で肩代わりするものである、と考えることができる。

VPN自体は、まだ発展途上の技術ではあるが、クライアント側でのencapsulation/decapsulationをユーザ機用OS（Windows等）で行うソフトウェアも開発途上ながらリリースされており、従ってモバイルユーザ側もマシン1台から構成できるため、この機能は比較的気軽に導入できる。

3.3.2 非標準ポートの使用

TCP/IPにおいて、同時に並行して発生する多数の通信を交通整理するとともに、その通信を担当するべきアプリケーションを識別するために、アドレス（IPアドレス）と併せて識別子として使われる番号がポート（或はポート番号）である。標準的なポート割り当ては、Well Known Portとして定められている。例えばPOPは110、SMTPは25である。多くのメールソフトウェアでは、上記のポートはデフォルト値として設定されているが、変更も可能であるケースが多い。

この双方のサービス（特にSMTP）を司るポート番号を、標準と別の番号にすることができる。このポート番号は、他のサービスとの衝突のない番号の中から、モバイルユーザとシステム管理者の間で合意のとれた番号である必要があり、さらに、SPAMmerから推測されづらい番号であることが望ましい。この非標準ポートでのMTAでは、外部から外部への中継要求を受け付ける。

ただし、外部から組織内へのメールを受信できるように、標準ポート25番でもMTAを（同一マシンでも別マシンでもいいから）動作させておく必要があるが、こちらの標準ポートMTAでは、外部から外部への中継要求は無条件に拒絶する設定で構わない。

ポート番号は16ビット整数で表現され、即ちその範囲は高々数万である。SPAMmerはしらみつぶし的に探索することによりこの非標準ポート番号を発見することはできるため、この対策は万全なものではない。しかし、次項の対策と併用することにより、モバイルユーザ（およびSPAMmer）からのコネクションとその他のコネクションを分離して、次項のプログラムの動作効率の低下を防ぐことはできる。

3.3.3 POPサーバとSMTPサーバの連携

Unixを用いたTCP/IPネットワークサーバーでは、一般に各プロトコル毎に別々のサーバソフトウェアが用意される。1プロトコル1サーバ、の関係がほぼ成立している訳である。同一のMUAからの要求を分担していようと、SMTPとPOPの関係においてもそれが成り立っており、両サーバは互いに無関係に動作する。しかし、サーバ機上に何らかの仕組みを準備し、両プロトコルによる通信同士の間で何らかの連携を行うようにすることは可能である。

多くのMUAでは、サーバ側スプールに溜まった受信メールのチェックと、MUA側送信待ち保管域に置かれたメールの受信処理とを一括して行う。POPとSMTPによる通信を（ほぼこの順で）逐次的に実行する訳である。この一般的傾向を利用して、POPによるユーザ認証が成功した後、当該POPセッションが終了した時点から、一定時間だけ、そのPOPセッション開設元のアドレス（からのコネクション）に限って、対外部中継要求（SMTPによる）を受理するように設定することにより、モバイルユーザへの便宜を提供しながら、なおかつSPAMmerからの中継要求をかなりの高い率で拒絶することが可能になるだろう。表現を変えれば、POPセッションの成立を「鍵」として、SMTPセッション開設のための「門」をその「鍵」で一定期間開いておく、という考え方である。

4 popKey（仮称）の実装

4.1 前提

前述のようにPOPによるアクセスの認証が成功したことをキーとして、当該クライアント機からのSMTPによるリレー要求を受理するゲートを一定時間だけ開く、という機能を実現する一対のプログラムを試作した。このプログラム群を仮にpopKeyと名づけておく。popKeyの実装にあたっては、以下の点を考慮した。

a この機能はPOPサーバおよびSMTPサーバへの機能追加である。研究の一環としてこれを実現する場合、オープンソースで流通しているソフトウェアをもとにした改良を手がけるのが妥当な選択であろう。さもないと著作権・ライセンス・技術情報といった厄介な問題にぶつかる可能性が高い。しかし、オープンソースによる有力ソフトの殆どはいまだ（或は永遠に？）発展途上であり、バージョンアップは続けられている。ここでの改良をそのバージョンアップと同期させないと、今後のバージョンアップの度になんらかの対応を行う必要が生じてしまう。従って、当該ソフトウェアをソースレベルで書き換えることを必要としない実装を行うことができれば、その方がより望ましいだろう。

b POPによるアクセスの認証が成功した際に、開門の鍵となる情報として記録し利用できる情報としては、アクセス元クライアント機のIPアドレスと、メールボックスのユーザ名が考えられる。現在、モバイルユーザが使うマシンの大半はWindows等のシングルユーザOS（複数のユーザが同時に使用できないという意味）であることを考えれば、上記の

うち、IPアドレスをキーとして使い、開門時間を適切に調節する、という方式で、ここでの目標に対して一応の効果は得られそうである。一方のユーザ名については、前述のようにメールの投函と転送の両方の要求に対するサーバとして実現されている現在の MTA が、発信元ユーザ名のチェックを厳しく行っていないことを考えるので、このユーザ名情報の利用のために MTA ソフトウェアへのソースレベルの追加がかなり必要になり、その割には効果の大幅な向上にはならないため、ここでは見送ることにする。つまりIPアドレスだけをキー情報とすることにする。

なお、MDA プロトコルとしては、現在 POP 3（POP のバージョン 3）が主流であるが、IMAP 4（同様）も普及しつつある。これらは、機能とにおいてはかなりの違いがあるものの、セッション開設時のユーザ認証過程はほぼ同じ形であるため、本章での説明は原則として POP に限定し、必要に応じて IMAP にも言及するものとする。また、具体的な MTA ソフトウェアとしては、古くからそして現在も広く使われている sendmail[14]と、最近注目されている qmail[15]とを想定する。

4.2 連携機構の検討

4.2.1 POP アクセスの認証成功事象の記録

POP によるアクセスの最初に、クライアントは USER コマンドと PASS コマンドまたは APOP コマンドを、必要なパラメータとともに送出する。それを受けてサーバはユーザ認証を行い、認証に成功するとそれ以降のコマンドが送出され、実際のPOPセッションが開始される。この過程は単一の TCP コネクション上で連続的に行われる。このユーザ認証が成功した時点で、クライアントのIPアドレスを、SMTPサーバから参照可能な形でサーバ上で記録する必要がある。POP サーバのソースプログラムを改変せずにこれを実現する方法としては、以下の3つの方法が考えられる。

a syslogへの出力の監視：一般にPOPサーバをはじめ多くのサーバは、イベントの発生をsyslog[16]のメカニズムを用いて記録する。POPサーバの場合、最低限、ユーザがログインに成功したというイベントは記録されることが多い。この syslog 記録要求を監視するデーモンプロセスをシステムに1つ常駐させるという実装も可能である。その際、

- syslogd に対して送られる UDP パケットを監視する
- syslogd が記録を残すファイルを tail -f で監視し、一定のパターンの行の出現を検知する

といった方法が考えられる。ただしこの方法による場合、ログイン成功のタイミングは検出できるが、POPセッション終了のタイミングを検出できない。POPセッションの所要時間は、ユーザの接続環境やメール利用行動（メールチェックの頻度や受信メールの量）により大きく変動するため、POPセッション開始時点から開錠時間をカウントするとすると、その変動を見込んで開錠時間をかなり長めに設定することが必要になってしまい、popKey の目的から考えると望ましくない。

b 透明なプロトコルフィルタ：一般に POP サーバはコネクション要求発生毎にinetd[17]等のスーパサーバから子プロセスとして起動され、スーパサーバから当該コネクションのためのソケットを標準入出力として継承する。この POP サーバ起動の過程で、この標準入出力ポートに対する双方向のフィルタとして働くプロセスを挿入し、そのプロセスがクライアントサーバ間の通信を監視するようにすることができる。間に挟まるプロセスはクライアント・サーバの双方からのメッセージを何も加工せずそのまま宛先ポートに送るが、その過程で通信の進行を監視し、ユーザ認証が成功したことを示す応答を認識させることができる。この種の双方向フィルタの実現のためには、pipe()、dup() といったシステムコールを活用したプログラミングが必要だが、POPのサーバからクライアントに向けての下り応答のみを監視し、

```
+OK user xxxx's maildrop has nnn
    messages (yyy bites)
```

というような応答に反応するような片方向

フィルタによる簡便なインプリメンテーションでも概ね問題はないようである。この場合は、シェルスクリプトでパイプラインを使って、

```
：
exec popd | some filter
```

というような呼び出し方で実現できる。

c 分離プロセス式サーバの利用：QMAIL に添付される POP サーバは、いくつかのプロセスのカスケード起動により実現されており、認証要求のコマンドを処理するプログラムと、セッション開始後のコマンドを処理するプログラム（これがサーバ本体となる）が別のプログラムで実現されている。OS（とりわけスーパーサーバ）の位置から見ると、後者のプログラムに前者のプログラムをかぶせる形で起動されることになるため、前者のプログラムを後者に対するラッパ（wrapper；外側から包むプログラム）と呼ばれることがある。これらのプログラムの呼び出し関係の間に別のプログラムを挿入し（2重、3重の入れ子の包装関係が成立する）、その中間プログラムに認証成功を記録させるという実装が qmail では可能である。

利用している MDA に応じて、b および c を選択するのが、差当たって手軽なアプローチだと考えられる。

4.2.2 SMTP サーバからの参照方法

SMTP サーバの最近の版（例えば sendmail）は、どの機械からのリレー要求を受理するかについて、受理すべき、或は拒否すべき要求元アドレスを列挙したファイルをアクセス発生毎に参照している。したがって、そのファイルを POP セッションの開設状況にしたがって自動的にリライトしていく仕組みを作ってやれば、SMTP サーバのソースレベルでの改良なしで、連携の仕組みが実現できる。

SMTP サーバが参照するのは 1 つのファイルである。このファイルに追加すべきレコードを発生させる POP セッションは 1 台のサーバ機上で非同期並行的に複数起動され得る。しかし Unix などの主要 OS においてはファイルへの追記は原子的操作として提供されていないため、各 POP プロセスからこのファイルに直接追記を行うと、タイミングによってはファイル内容の一貫性が失われることになる。これを防ぐために何らかの同期機構が必要である。

a ファイルシステム上にロックファイルを作るなどの方法によりプロセス間の排他制御を行う

b デーモンとして常駐するプロセスを 1 つ置き、このプロセスに対するリクエストを送ることにより同期をとる（前項の syslog を用いる方法の場合、syslogd がそのプロセスの役割をはたすことになるが）

このうち、a の考え方に基づく方法の場合、クリティカルな資源のロックが解除されるのを待つために POP ラッパ内で時間待ちのループを作る必要がある。この時間待ちのためにファイル更新が SMTP セッション開始に間に合わなくなる恐れもあり、望ましい方法とは思えない。

ここでは上記の b をベースにすることにした上で、もう 1 つ決定する必要があるのは、POP セッション開設状況をデータとして保持するためのデータ構造である。これについては、単一のディレクトリを使うことにする。ディレクトリエントリの生成・削除（ファイルを作ったり削除したり）は、原始的操作として提供されているため、このデータ構造へのアクセスには排他制御の必要がないからである。このデータ構造（本稿のケースでは中間的データ構造と位置づけられる）は、次項のケースでは直接参照できるが、本項のように SMTP サーバが単一ファイルを参照することを条件としているケースでは、中間的データ構造から単一ファイルへの変換（大雑把に言えば ls＞someFile のような操作）を適切なタイミングで行うデーモンプロセスを用意する必要が出てくる。この操作を行うタイミングは、POP サーバ（のラッパ）が開門（ディレクトリエントリを作成）した直後であればいい。このタイミングを通知するためにはシグナル（下記の例では USR1 シグナルを想定）を使うことができる。以下に実装の大枠を示すように、このデーモンは、適当な時間間隔で有効期限切れ開門データ（ファイル）の除去を行う以外は殆どを寝て暮らすが、シグナル

を受信した時には目覚めて、データ変換処理を行う。なお、以下の例はシェルスクリプトの記方で示したが、実際には次節で述べるようにstat( )[18]の機能を持つ言語を使って実装する必要がある。

```
#!/bin/bash
sub convert { #シグナル受信時に呼ばれる
              #プログラム片
  :データ変換処理と次の休眠時間$sleepの
  :決定
}
seeep=600
pidFile=/var/run/popKey.pid
echo $$ > $pidFile
trap convert USR1
while true ; do
  :有効期限を過ぎたファイルの除去
  sleep $sleep
done
```

4.2.3 QMAILの場合

sendmailがメールに係わる多くのことを処理する1つの大きなプログラムとして実装されているのに対し、qmailで（そのアンチテーゼの意味もあり）、小さなプログラムの集合体として実装されている。その中で、SMTPによるリクエストを処理するプログラムは、qmail-smtpd[19]と名付けられている。sendmailが自分自身でSMTPポート（標準で25）をlistenしたり子プロセス生成したりする作業をすべて自分で行うのに対して、このqmail-smtpdは、スーパサーバ（inetdなど）から（大抵はラッパを何枚か被せて）呼び出されるように作られているため、ラッパの構成により柔軟に付加機能を実装することが可能である。

実際、中継要求の諾否についても、前項のようにsendmailは自身でファイル参照を行っているが、その代りにqmail-smtpdは、環境変数RELAYCLIENTが設定されているかどうかをチェックすることでこれに代えている。ファイルを参照してこの環境変数を設定する作業は、qmail-smtpdのラッパとして標準で用意されているtcpd[20]が行っている。そこで、POPセッションの開設状況をチェックする機能は、

a tcpdに代るラッパとして

b tcpdとqmail-smtpdの間に挟まるラッパとして

のいずれかの位置で実現できるが、tcpdが実際にはもっと多機能な（事実、qmail-pop3dに対するラッパとしても使われている）プログラムであることを考えると、プロセス生成のオーバヘッドが気になるような性能的に非常に緊迫した状況のサーバでない限り、bのアプローチで充分だと考えられる。この場合、開門状況を示すデータ構造は、単純なファイルである必要はないため、前項で（は中間データ構造だったが）示したディレクトリエントリによるデータ表現が、そのまま使える。

4.2.4 MUAと開門時間について

MUAがMDAプロトコルのセッションとSMTPによるセッションとをどのような順序で行うか、については、特にどこか既定がある訳ではなく、MUA作成者に委ねられている。従って、たまたまここで前提としていない順序で通信を行うようなMUAも存在するだろう。その場合、そのMUAはモバイルユーザ向けのMUAとしては本方式では使用できないことになる。

次節の例では、微妙なタイミングでSMTPセッション開設に失敗することをさける目的で、POP認証成立後すぐに（即ちPOPセッションが開いている期間から）開門を行っている（ただし開門時間の計測はPOPセッション終了時点を起点にしている）。この方式では、POPとSMTPの通信が並行して行われるようなMUAでもある程度使える可能性がある。POPにおいては、そのセッションの継続時間内にユーザが介入する余地は殆どありえないため、スプールに溜まっていたメールをまとめて転送するための機械的時間であり、POPセッション開設中からの開門はさほど危険ではないだろう。一方、IMAPについては、まだその機能を充分に活用する決定的なMUAが出現していないという現状であり、今後のどのように発展するか未知数なのだが、インタラクティブに使用される可能性は多々あり、その場合、IMAPの（機械的意味における）セッションは長い時間継続する可能性があることを念頭においておきたい。

4.3 QMAILのための実装の詳細

4.3.1 POPサーバ側

一般的には、ラッパプロセスは、ラップされるプログラムに対する前処理だけを行うため、以下のような大筋でexecシステムコールにより自分自身が次のプログラムに変身することにより被ラッププログラムを起動する。シェルスクリプトの枠組みで以下のように表現できる。

```
前処理
prog=$1；shift；exec $prog "$@"
# not reach here
```

しかし、このpopKeysの場合は、POPセッションの終了を記録する必要があるため、execを使わない。つまり普通のコマンド呼び出しと同様の起動方法を用いることになる。

```
前処理
prog=$1；shift；　　$prog "$@"
後処理
```

実際には、（さらに省略化した記法を使うが）すべての処理を書いても以下のように数行程度のプログラムでPOPサーバに対するラッパを実装できる。

```
#!/bin/bash
：変数設定
$ctlDir=/var/run/popKey
$peerIPaddr=
$genName=$workDir/$peerIPaddr
$uniqName=$genName+$$
：前処理
touch $uniqName
：被ラッププロセス呼出し
"$@"
：後処理
touch $uniqName；mv -f $uniqName $ipAddr
```

ここで、環境変数TCPREMOTEIPは、この上位のラッパであるプログラムtcp-env[21]が設定した、相手方のIPアドレス（ドット表記したもの）である。ディレクトリ$cdlDirにそのIPアドレスを名前とするファイルが存在していることにより「開門」を意味させる。$uniqNameはPOPセッション開設中を示す記号として使われる。$uniqNameでは、同一IPアドレスから同時に複数のPOPセッションが開いた場合のファイル名の重複を（POPセッション開設中に限って）避けるために、プロセスIDを付加している。プロセスIDは循環使用されるため完全に重複をさけられる保証はないが、これらのファイルの生存期間はプロセスIDの循環の周期よりはるかに短いと考えられるため、それ以上の複雑な名前生成は行っていない。

このプログラムをpopKeyと名づけ、/usr/sbinに置くとすると、スーパサーバから

```
/usr/sbin/popKey qmail-pop3d
```

という形で（さらに上位のラッパはここでは省略した）呼び出すよう設定すればいい。

4.3.2 SMTPサーバ

一方の、SMTPサーバからこれを参照するためにSMTPサーバに被せるプログラムだが、最低限必要な機能は、

- 当該IPアドレスを名前とするファイルが存在し、
- それが有効期間内であるか

を調べて、この条件が満たされたら環境変数RELAYCLIENTを設定した上でSMTPサーバを呼び出すことである。このためには、当該ファイルについて最終更新時刻からの経過時間を秒単位で調べる必要がある。この機能はシェルスクリプト（またはそこから容易に呼び出せる機能）では実現できず、システムコールstat()またはそれに代る機能を提供する言語を用いて（全部または当該機能のみを）実装する必要がある。筆者はperl言語[22]を用いた。

また、ディレクトリに古いエントリが数多く溜まるのは望ましくないため、古いエントリは適宜消去することも必要である。これについては、cronから、適当な間隔（例えば1回/日）でfindコマンドを呼び出す方法でも構わないが、以下の例では、その機能も組み込んだプログラムを示す。

```
#!/usr/bin/perl
$gateDur=10；$xpireDur=300；
$ctlDir='/var/run/popKey'；
$curTime=time；
$peerIPaddr=$ENV {'XXX' } ；
chdir $ctlDir；
opendir (DIR,'.')||die"opendir：$!\n"；
```

```
for (readdir DIR) {
  /^\.+$/&& next ; # ignore .. & .
  $fAge=$curTime - (stat ($_)) [9] ;
  if ($fAge>(/\+/?$xpireDur : $gateDur))
  {
     push (@xpired, $_) ;
  } elsif(/^$peerIPaddr(\+\d+)?$/){
     $ENV { 'RELAYCLIENT' } = 1 ;
  }
}
closedir (DIR) ;
unlink @xpired ;
exec $ARGV ;
```

なお、上記のファイル削除機能を省略し、POPセッション終了後に初めて開門するようにした簡易版の場合、以下のように実装できる。

```
#!/bin/bash
file=" /var/run/popKey/$XXX"
if test -f $file &&
    'perl -e 'print time- (stat (\"$file\"))
     [9] ; ' '-le 10
then RELAYCLIENT= 1
      export RELAYCLIENT
fi
exec"$@"
```

5 考察

5.1 コミュニティの変容

以前から計算機科学の領域は kill, abort, execute といった不穏な用語の飛交う、怪しげな世界ではあったが、それらはあくまで比喩としての用語であった。純粋に技術者として生きていたつもりの筆者が「動機は怨恨」「犯罪を構成」といった、三面記事的（或は刑事ドラマ的と言うべきか）用語を含む論文を書くことになるとは予想していなかった。

この根源には、本稿初頭で述べたようなインターネットの変容がある。かつてはインターネットを利用するのは、大学や大企業といった特殊な立場にいる人間に限定されていたが、現在では多少のコストを支払うだけで誰でも容易にアカウントを手に入れることができる。インターネットは、よりオープンな世界になった。これは裏をかえせば、悪意の人間もそこに入り込み、また活動することが容易になったということでもある。素性の知れた面々で構成される「村」から、誰かわからない不特定多数で構成される「都会」への変質であるともいえる。

その中で、自身の安全を確保するために、交際範囲（メールを受け取ることも含めて）を限定してインターネットの内部、或は外部に、部分的にでも閉じたコミュニティーを形成しようとする動きが生じるのは、当然の成り行きかもしれない。いわば「村」への回帰である。例えば前述のRBLは、良質なシステム管理の努力を怠らないサイトだけに交際範囲を限定するための機能であると言えるだろう。また、現在のインターネットとは別のネットワークを（つまり外側に）新たに構築しようという動きが「インターネット2」[23]「NGN」[24]などのプロジェクトとして進められており、これは研究環境確保がその主たる目的ではあるが、「都会」の煩わしさを裂けることもまた、その隠れた動機の中に含まれているであろうことが想像はできる。

また、第1章でインターネットのコスト負担モデルについて論じたが、これは、インターネット上の諸資源は、「村」的コミュニティの中で理性的なユーザにより良心的にシェアされることを暗黙の前提として形づくられてきたものであることを考えると、「都会」においては、そのコスト負担も、別の形態で行われるのがふさわしいのかもしれない。それは現実世界における「都会」と同様に、殺伐としたものになる可能性もあるのだが。

5.2 サーバの実装のための環境について

5.2.1 OSの機能

ここで検討を行った実装レベルの議論においては、非同期に発生する複数のプロセス同士が、安全で確実な連携を行うための、手軽な枠組みを、Unixにおいて（或はそれに類するサーバ用OSにおいても）提供していないことが話を複雑にしていることが（特に4.2.2項あたりで）見てとれるだろう。本項のテーマとは別に、sendmail（或はそれを前提としてプログラム群では）ユーザ毎のメールスプールを1つのファイルとして実現しており、このことは内発的脅威への耐性のなさや、

ソフトウェアが徒らに複雑になるなどの、様々な問題の原因となっている。

前章での議論を考えると、ディレクトリ、ファイル、そして、プロセス、の3種類のものを別々のセマンティクスで扱う必要があるという点で、インターネットで主流として使われるサーバは、古典的なOSで構成されている、と言えるのかも知れない。前章での議論でいえば、「門」の機能を

・データの「コレクション」（一般にオブジェクト指向で使われる用語で表現した場合）のサブクラスで
・レコード単位の安全な追加と削除、と
・レコードの検索をするためのメソッドを提供し、また
・古いレコードを自動的に破棄する機能を持つ

ようなオブジェクトとして実現することにより、サーバソフトウェアの実装が非常に楽になるだろう。本格的なオブジェクト指向OSの出現が待たれるところなのだろう。

5.2.2 言語の機能

OSと並んで、開発言語もまた環境として重要な要素である。本稿で示した実装例は性能よりも短期の開発を重視して、シェルやperlといった高レベルの言語を用いている。こういった言語の存在が、システム管理者に大きな恩恵として働いてきていることは、ここでの経験においても痛感させられた。

一方、サーバソフトとして実用に供されてきた多くのプログラムは、これまで殆どC言語で書かれてきている。C言語は後の言語に大きな影響を与えた優れた言語ではあるが、いかんせん入出力まわりに大きな弱点を持っている。（実際にはC言語の標準ライブラリが持つ弱点ではあるが）。実際、C言語により実装されたサーバソフトで、入力バッファのオーバーフローに起因するセキュリティホールが、例年のように報告されている。言語の機能を正しく使えば発生せずにすむセキュリティホールなのかも知れないが、正しく使いきれずに些細やチェック忘れや誤りを誘発してしまうような、言語独自の特性があるのかも知れない。開発言語についても、次の時代がそろそろ来るのかも知れない。

5.3 今後の課題

本論文では、止むにやまれぬquick hackを行った第4章部分を除いては、殆どが机上の空論であるとも言える。ここでの検討内容や提案事項を実際にインプリメントすることにより、実証を試みることが今後の課題となるだろう。また、勿論、ここでの検討をこの場所だけのものに留めておかず、インターネットコミュニティに投げかけて行き、より安心なネットワークが実現することに何らかの貢献をすることも、また筆者にとって重要な課題だと考えている。

（1999. 1. 30　受理）

謝　辞

本学のネットワークに関して、様々なレベルで情報を提供して頂き、また結果として研究題材を提供して下さる情報システムセンター関係者各位に、また、筆者にテストベッド的環境を提供して下さる、（事情により名はあげないが）企業の関係者各位に、感謝を申し上げる。また、本稿の第3、4章で述べた具体的対策の実施には、qmailに関するメーリングリスト[25]で行われている議論等をヒントにさせて頂いたことも付記し、関係者各位にお礼を申し上げる。

注

1）http://www.masternet.or.jp/inter/10mail.html
2）平岡信之「キャンパスネットワークにおける資源の有効利用（I）」長野大学紀要、15(4)、16-32、1994
3）平岡信之「キャンパスネットワークにおける資源の有効利用（II）」長野大学紀要、16(1･2)、25-40、1994
4）J.B. Postel, “Simple Mail Transfer Protocol”, RFC-821, 1982
5）J.Myers, et.al., “Post Office Protocol”, RFC-1939, 1992
6）M.Crispin, “Internet Message Access Protocol”, RFC-1996, 1996
7）http://caramia.g-net.org/spam/
8）http://caramia.g-net.org/spam/whatis.html
9）http://maps.vix.com/rbl/
10）http://www.jp.qmail.org/rbl/rblsmtpd/
11）http://www.pgpi.org/
12）R. Fielding, et.al. “Hypertext Transfer Protocol ——HTTP/1.1”, RFC-2068, 1997

13) R.Gellens, et.al., "Message Submission", RFC-2476, 1998
14) http://sendmail.org/
15) http://www.jp.qmail.org/
16) syslog(8), FreeBSD-2.2.6：以下、この書式の文献参照は、Unix オペレーティグシステムの man コマンドにより参照するオンラインマニュアルページ（カッコ内はその章番号）および、筆者が参照したデータが添付されていた OS またはパッケージ名を示す。
17) inetd(8), FreeBSD-2.2.6
18) stat(2), FreeBSD-2.2.6
19) qmail-smtpd(8), qmail-1.0.1
20) tcpd(8) FreeBSD-2.2.6
21) tcp-env, (8) qmail-1.0.1
22) http://www.perl.com/
23) http://www.internet2.edu/
24) http://www.ngi.gov/
25) http://www.jp.qmail.org/ml/