第 11 部

ファイアウォール構築技術

第 1 章

プライベートアドレスの利用と問題点

1.1 ファイアウォールとプライベートアドレス

ファイアウォールの実装にはさまざまな方式が考えられるが、本節においては特にファイアウォール内側のネットワークにプライベートアドレスを割り当てる方法を用いた場合に想定される問題点について検討する。

プライベートアドレスは初め RFC1597[82] において IANA による予約が宣言され、現在は RFC1918[83] に定められているアドレス空間で、インターネット上での経路広告が行なわれないことを前提としたネットワークに対して割り当て可能なネットワークアドレスを指す。特別の割り当て手続きを必要とせずに組織内部のネットワークに割り当てて利用することができ、インターネットに対する直接の接続性を持つ必要がないネットワークが組織内部に存在する場合に、利用が推奨されている。

インターネットに対する直接の接続性が必要ないネットワークにおいては、そのネットワークに対する経路情報をインターネット上に広告する必要はない。従ってそのネットワークと同じアドレスのネットワークが他の組織内に重複して存在していたとしても、運用上差し支えない状態に設定して維持することが可能である。

インターネットへの経路情報の広告が必要ないという前提の元では、組織内に閉じたネットワークに対しては次のいずれかのネットワークアドレスを割り当てて使用することになる。

1. 正式に割り当てを受けたポータブルアドレス
2. 正式に割り当てを受けた CIDR ブロックアドレスのサブネット
3. プライベートアドレス
4. 正式に割り当てを受けていないアドレス

これらのうち、1. ～3. は正式に使用可能なネットワークアドレスである。それに対し 4. は正式には使用可能でないネットワークアドレスであり、本来使用することは避けるべきである。しかしながらある種の事情により使用せざるを得ない場合があり、また安全に運用することも不可能ではない。この場合の詳細に関しては 2 を参照のこと。

正式に使用可能なネットワークアドレスのうち、1. はインターネットに対して経路情報を広告可能ではあるが、インターネット全体の経路情報の数を減らすためには広告しない方が望ましい。さらに広告しない場合にはアドレス自体の IR [1]への返却が要請されている。また 2. は既にインターネットに対して経路情報が広告されているネットワークであり、これを組織内ネットワークに対して使用すると、広告される経路情報の数に対してインターネットからの接続性が有効となるホストの数が少なくなるため、アドレスの利用効率の観点から使用することはあまり推奨されない。

以上の理由から、最近では新規にインターネットへの接続を行なう組織では、ファイアウォールの内側に設置する組織内のネットワークに対して、プライベートアドレス空間からネットワークアドレスを割り当てることが一般的となっている。

プライベートアドレスとして利用可能なネットワークアドレスは、次のいずれかのブロックに属するアドレスとなる。

```
10.0.0.0      ～  10.255.255.255   (10/8)
172.16.0.0    ～  172.31.255.255   (172.16/12)
192.168.0.0   ～  192.168.255.255  (192.168/16)
```

この範囲内のネットワークアドレスは、割り当て申請などといった特別な手続きを経ずに自由に使用することができる。また複数のブロックから選んだアドレスを組み合わせて使用しても構わない。

組織内でネットワークに接続されているホストは、次の 3 種類に分類することができる。

1. 組織外のホストとは一切通信しないホスト
2. 組織外のホストとの通信を限定した形で行なうホスト
3. 組織外のホストとの間でネットワーク的な接続性を必要とするホスト

ここで 1. と 2. を合わせてプライベートホスト、3. をパブリックホストと呼ぶ。

1. のプライベートホストは、全ての通信が組織内で閉じたものとなるホストである。それに対して 2. のプライベートホストは一般に大部分の通信が組織内で閉じるが、電子メールや WWW、ファイル転送などといった特定のネットワークサービスに限定して組織外との通信を行なう可能性があるホストを指す。この両者はときにより明確に分けることができない場合がある。例えば通常は 1. として運用されているが、特定の状況下でのみ 2. として運用されるホストなどが存在し得る。

2. における組織外との限定した形の通信がどのように行なわれるかは、その組織が組織外のインターネットと接続する部分のファイアウォールの実現方式に依存する。また組織外のどの範囲のホストに対してどのような通信を行なうかは、ファイアウォールの運用方

[1]Internet Registry: IANA から委譲を受け、IP アドレスなどの割り当て業務を行なう組織。regional NIC, country NIC などを指す。

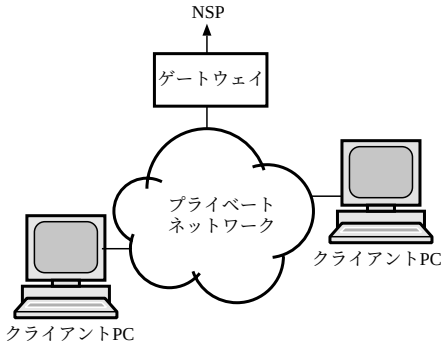

図 1.1: ファイアウォール構成例 (1)

表 1.1:

構成例	パブリックホスト	プライベートホスト	広告する経路
(1)	ゲートウェイ	クライアント PC	ゲートウェイ
(2)	外向けルータ 外向けサーバ	内向けルータ クライアント PC	バリアネットワーク

針に依存する。ここでは、プライベートホストが接続されているネットワークに対してプライベートアドレスを割り当てる形式のファイアウォールに限定して考えることにする。

プライベートアドレスを使用する場合の、一般的なファイアウォール構成を単純化すると、図 1.1のような構成か、図 1.2のような構成と捉えることができる。いずれもプライベートネットワークにはプライベートアドレスを割り当ててある。両者を簡単に比較した結果が表 1.1であるが、最も大きな差は広告する経路の情報である。構成例 (1) ではゲートウェイに対するホスト経路のみの広告で良いのに対し、構成例 (2) ではバリアネットワークに対するネットワーク経路を広告する必要がある。特にインターネットに対し、ネットワークサービスをサーバとして公開する必要がなければ、構成例 (1) においてはゲートウェイの NSP 側アドレスはダイアルアップ PPP 接続で動的に定まる IP アドレスであっても運用は可能である。

上記のようなネットワーク構成において、プライベートアドレスを利用する場合のメリットをまとめる。

リナンバリングの必要性が少ない

一般に組織内のネットワークアドレスを変更する作業 (リナンバリング) はインターネット接続時や接続 ISP 変更時に行なわれる。

組織内のネットワークを設計する場合に、最初からプライベートアドレスを利用する

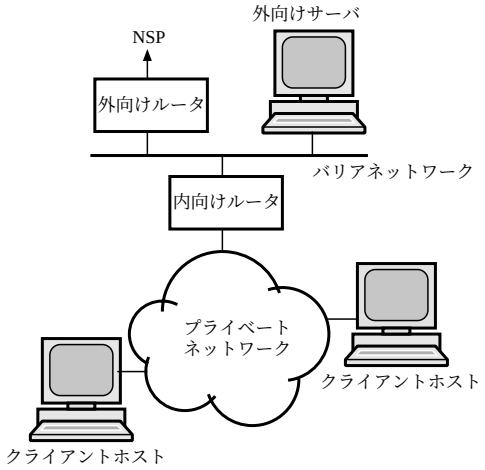

図 1.2: ファイアウォール構成例 (2)

ように考えていれば、運用を開始してから後にリナンバリング作業を必要とする機会が減らせる。
また既にポータブルアドレスや正式に割り当てを受けていないアドレスを組織内で使用してしまっている場合には、いずれどこかの時点でリナンバリングする必要がある。その際にプライベートアドレスへ移行することにより、大抵の場合には面倒なリナンバリング作業を一度で済ませることができる。
リナンバリング作業に関しては、RFC1900[84] や RFC1916[85] を参照のこと。

経路情報増加の抑制に貢献する

組織内で利用されるホストの大部分に対してプライベートアドレスを割り当てることにより、インターネットに対して必要最小限の経路情報を広告するだけで済むため、近年著しいインターネット上の経路情報の増加を抑制する効果が期待できる。

IP アドレス資源を有効活用できる

組織内で利用されるホストの大部分に対してプライベートアドレスを割り当てることにより、一部のパブリックホストにのみインターネットから到達可能な IP アドレスを割り当てれば済むため、近年著しい IP アドレス資源の枯渇状況を緩和する効果が期待できる。
なお既にポータブルアドレスを取得して使用している組織がプライベートアドレスへの移行を行なった結果、使用されなくなったポータブルアドレスを RFC1917[86] に従い IR に返却することが強く推奨されている。

ネットワーク構成の自由度が高い

現在 IP アドレス資源にゆとりがないため、各組織に対して IR より割り当てられる

ネットワークアドレスは必要最小限の数だけに限定されている。その中で組織内の各サブネットに適切な数のアドレスを割り当てるためには、サブネットごとに接続されるホスト数の違いなどから可変長サブネットマスクの使用が必須となる組織が多い。また将来のホスト数増加を見越してアドレスを割り当てることは困難を極める。
こういった点は、ネットワーク構成を設計する際の難度を高めたり、アドレス割り当てに要する管理の手間を増やしたりする要因となる。
プライベートアドレス空間には、組織内での割り当てを行なうためには十分な数のアドレスブロックが用意されているため、プライベートアドレスを使用することにより割り当て管理の負担を抑えることができる。

正式に割り当てを受けていないアドレスを使わずに済む

現時点においてインターネットとの接続を行なっていないが、将来的に行なう予定が少しでもある組織に対して、従来はポータブルアドレスの割り当てが行なわれていたが現在は行なわれない。そのような組織が使用するアドレスとして、プライベートアドレスは適切である。

一時的なネットワークを作りやすい

1.3 を参照。

逆に、プライベートアドレスの利用により次のようなデメリットを考えることができる。

ネットワークサービスの利用が制限される

1.2 を参照。

複数のネットワークを統合する時のアドレス衝突

1.4 を参照。

より高度な要求に対応しにくい

プライベートアドレスを使用したファイアウォールにおいては、基本的に組織内に閉じた利用と組織外へのアクセスの切り分けがネットワーク単位になってしまうという問題がある。
例えばプライベートホストとパブリックホストの密な混在や、同一ホスト内でのユーザ単位/サービス単位での切り分け、あるいは状況に応じた動的な切り替えなどは困難と言わざるを得ない。

経路情報などの流出に注意する必要がある

組織内のネットワークにプライベートアドレスを使用している場合、組織内の経路制御を動的に行なっているときには、プライベートアドレスのネットワークに対する経路情報が組織外に流出しないように細心の注意を払う必要がある。
またソースアドレスやソースルーティングアドレスにプライベートアドレスが含まれ

たIPデータグラムや、プライベートアドレスが記述されたDNSのリソースレコードなども組織外に流出しないように注意する。特にMXホストがファイアウォールホストを兼ねているなどの場合に、インターネットからアクセス可能なアドレスと組織内で使うプライベートアドレスの両方をAレコードとして登録してあると、たまたま同じプライベートアドレスを利用している組織からのメールが届かなくなるなどのトラブルが生じ得る。

1.2 ネットワークサービスの制限

一般にファイアウォールは、ネットワークサービスの利用に対する自由度とひきかえに安全性の高さを得るための機構と考えられる。そこには双方の要求に対するある程度の妥協を想定する必要があるが、この妥協点をより現実的なものとするために、ファイアウォールの設計方針を十分に検討することは重要である。

組織内のネットワークにプライベートアドレスを割り当てる方式のファイアウォールにおいて、組織外のネットワークサービスを利用するためにトラフィックを中継させる機構としては現在次のような実装が実用化されている。

- Proxyサーバによるアプリケーションレベルの中継
- SOCKSサーバによるトランスポートレベルの中継
- NATルータによるネットワークレベルのアドレス変換

これらはまた、混在して利用することも可能である。

ここでは上記各方式の特徴と得失を比較し、プライベートアドレスを使用するファイアウォールを構築する際に検討すべきネットワークサービスの利用制限を考察する。

初めに、Proxyサーバを利用する方式の特徴および得失を考えてみる。

Proxyサーバを利用するクライアントには、Proxyサーバの存在を意識するものと意識しないものがある。クライアントを使用するユーザにとっては、それらはクライアントアプリケーション上にProxyサーバの設定項目があるかないか、あるいは組織外へのアクセスをする際に一旦中継所にアクセスすることを意識するかしないかの違いとして認識される。

Proxyサーバの存在を意識するクライアントにおいては、通常Proxyサーバを設定する機能が用意されている。クライアントは組織外へのアクセスにおいて、常に自動的にProxyサーバ経由でのアクセスを行なうようになる。ユーザは一度Proxyサーバを設定しておけば、以後Proxyサーバの存在を意識する必要はない。

Proxyサーバの存在を意識しないクライアントにおいては、組織外へのアクセスを直接行なうことはできない。ユーザは組織外へのアクセスを行なうための中継所として、毎回意識してProxyサーバへのアクセスを行なうことになる。

前者の例としては、多くの WWW ブラウザや RealAudio クライアント、StreamWorks クライアントなどが存在する。それらに対し、各々HTTP Proxy サーバや RealAudio Proxy サーバ [2]、StreamWorks Proxy サーバ [3]などといったサービスごとの Proxy サーバが用意されており、中継を行なうパブリックホスト上ではその都度使用する Proxy サーバをインストールする必要がある。

後者の例としては、DeleGate [4]や fwtk [5]のような汎用の Proxy サーバを使用することができる。前者と比較して後者の場合には、Proxy サーバに対応した特別なクライアントを用意する必要はなく、telnet や ftp といった標準的なクライアントソフトウェアをそのまま使用することができる。

またユーザが直接使用するクライアントソフトウェア以外に、メール中継システムやニュース配送システムのようにバックグラウンドで組織外とのネットワークサービスを中継するソフトウェアも存在する。こういったソフトウェアは本質的に情報を中継することを前提として設計されたシステムなので、Proxy サーバ方式として扱うのは適切とはいえないが、動作としては近いと考えることができる。

次に SOCKS サーバを利用する方式の特徴および得失を考える。

SOCKS サーバを使用するためには、SOCKS に対応した専用のクライアントを用意する必要がある。ユーザから見ると使用上は、標準的なクライアントと透過に見えるのが SOCKS クライアントの大きな特徴である。そのため特別な設定項目や、使用上の注意などをユーザに教育する必要がない。

SOCKS サーバのインストールは一度行なえば良いが、クライアントは必要なものを全プラットフォーム分だけ用意する必要がある。また一般に入手可能な SOCKS 対応のクライアントソフトウェアは多くなく、特に市販のアプリケーションとしては皆無に等しい。ソースが入手可能なフリーソフトウェアのうち、単純な socket 接続による通信を行なうクライアントのみが利用可能という状況に近い。

ただし一部には Macintosh 用 FTP クライアントである Fetch [6]のように元から SOCKS に対応したクライアントソフトウェアが入手可能であったり、SOCKS 対応の wsock32.dll [7]のようにインストールするだけでネットワークアプリケーションが全て SOCKS サーバを利用できるようになる Windows 用の DLL が入手可能であったりするので、一概に全て駄目だというわけではない。ユーザに提供するネットワークサービスの範囲によっては SOCKS サーバを利用するファイアウォール環境を構築することは可能であり、また構築に成功すればユーザからは透過的に組織外が見えるため以後の運用が楽になる可能性も高い。

最後に NAT ルータを利用する方式の特徴および得失は以下のようなものとなる。

[2] http://www.realaudio.com/intranet/firewall.html
[3] ftp://ftp.xingtech.com/pub/streamworks/xdma_gw.c
[4] http://www.etl.go.jp:8080/etl/
[5] http://www.tis.com/Home/NetworkSecurity/Firewalls/FwtkOverview.html
[6] http://www.dartmouth.edu/pages/softdev/fetch.html
[7] http://www.hummingbird.com/press/socks.html

NAT ルータはアドレス変換機構を持ったルータで、ルータの片側にいるあるホストが、他方から見た場合に異なる IP アドレスを持っているかのように見せることができる。NAT ルータを利用すれば、組織内でプライベートアドレスにより運用しているホストが、組織外から見るとインターネットから到達可能な IP アドレスが割り当てられているホストであるかのように見せられるようになり、特別な中継機構を必要とせずにそのホストから組織外への通信が行なえるようになる。

この方式では特別なクライアントソフトウェアを用意したり、ネットワークサービスごとに中継サーバをインストールしたりする必要がないため、ユーザから見た利用可能性と管理者に掛かる管理コストを低く抑えることができる。しかしながら、NAT 機能を持つファイアウォールソフトウェアや NAT 専用ルータの導入が必要となる。

Proxy サーバ方式や SOCKS サーバ方式においては、組織外から到達可能であるべきホストはそれぞれ Proxy サーバあるいは SOCKS サーバが稼働しているパブリックホストだけで良いのに対し、NAT ルータを使う方式の場合には、組織外との通信が必要になる可能性があるホストの数だけインターネットから到達可能な IP アドレスを必要とするため、余裕を持った量の経路情報を広告しなければならないのは NAT ルータ方式のデメリットといえる。

また NAT ルータには、変換すべきアドレスを静的に定義する方式とその都度必要なアドレスを動的に割り当てる方式があるが、前者はホストを追加するたびに登録が必要になる点、同時に組織外との通信を行なうホスト数より多くの到達可能な IP アドレスを消費する点などが問題となる。逆に後者は実装が難しく、価格が高い点、動作が複雑になりトラブルが発生しやすくなる点などが問題といえよう。

以上の各方式の得失をまとめると表 1.2のようになる。これからも分かるように、全てのネットワークサービスを自由に利用可能な方法というのは実現が難しいか、あるいは現時点では不可能である。市場の要求に伴い、Proxy サーバや SOCKS サーバに対応するクライアントアプリケーションも増えてはいるが、新しいネットワークサービスがそういった機能を持つようになるにはやはり数ヵ月以上の時間が掛かる。WWW ブラウザから起動されるヘルパーアプリケーションやプラグイン機能、あるいは Java アプレットなどを使用したネットワークサービスについては、サーバと直接通信する必要がある一部の例外を除き HTTP に対する Proxy サーバを用意することにより利用可能になるが、それが新しいネットワークサービスの研究開発に枠をはめるようでは本末転倒といえる。クライアント側のことを考えなくて良いという意味では NAT ルータ方式が理想に近いのだが、運用実績などはまだ十分とはいえず、今後の普及に期待せざるを得ない。

いずれにしろどの方式がもっとも優れていると判断することは難しく、実際は設置可能な機種や管理に必要な工数、ユーザ教育の手間などを全体的に勘案した上で、複数の方式を組み合わせることなども含めて妥協点を探す努力が必要とされる。

表 1.2: 中継機構の方式による得失

方式	中継機構	クライアント
Proxy サーバ方式		
Proxy 設定あり	ネットワークサービスごとに、Proxy サーバをインストールして設定する必要がある。	Proxy サーバ対応のクライアントが必要。一度クライアント側で Proxy サーバを設定すれば、以後は Proxy サーバを意識する必要がない。
Proxy 設定なし	ネットワークサービスごとに、Proxy サーバをインストールして設定する必要がある。	特別なクライアントは必要ない。
SOCKS サーバ方式	SOCKS サーバをインストールして設定する必要がある。ネットワークサービスごとの設定は必要ない。	SOCKS サーバ対応のクライアントが必要。一度クライアント側で SOCKS サーバを設定すれば、以後は SOCKS サーバを意識する必要がない。
NAT ルータ方式	NAT ルータの導入と設定が必要となる。ネットワークサービスごとの設定は可能であるが必須ではない。	特別なクライアントは必要ない。

1.3 一時的なネットワークの敷設

プライベートアドレスは正式なアドレス取得のための手続きや経路情報の広告に要する作業、コストなどを考える必要がなく、手軽に利用することができる。そのため例えば比較的小規模で、かつ限定したネットワークサービスのみが利用できれば良い仮設ネットワークを設置する場合などの利用に適している。

インターネットとの接続に対しては、マルチホームルータを利用した単一ホストのみからなるファイアウォールを設置する。クライアントからは Proxy サーバを利用するネットワークサービスのみを使えれば良いとして、ファイアウォールホスト上で Proxy サーバを設定することでインターネット上のネットワークサービスに対するアクセスを行なうことができる。この場合、例えばクライアントからは WWW ブラウジングのみが可能であるなどのような制限が生じることになる。

Proxy サーバを使用するため、インターネットから到達可能となるホストはファイアウォールの一台のみで良い。したがって例えば商用 NSP のダイアルアップ PPP 接続サービスなどを利用すれば接続性を確保することができ、一時的なネットワークを敷設するごとに専用の経路情報を広告する必要は多くの場合ない。これはまた一時的なネットワークシステム自体がポータブルであることを意味し、一時的なネットワークを構成する機器類の設定をほとんど変更することなく、アクセスポイントを渡り歩きながらいろいろな場所を巡業することができるという側面も持つ。

以下に、上記の方針に沿った運用を行なっている小規模なネットワークの事例を紹介する。

小規模ネットワークにおけるファイアウォールの構築

ここでは 10 台以下の小規模なネットワークにおけるファイアウォールモデルを考え、モデルに沿った形でファイアウォールシステムを構築し、実際に運用した結果について報告する。

小規模ネットワークのモデルとして小規模ネットワークモデル SDI-1 (Sheep Defender for Internet 1) を考える。

SDI-1 の目的

SDI-1 は街のインターネットカフェなどで見られるような小規模なネットワークであり、喫茶店の客 (=不特定多数のユーザ) に対してインターネットのサービスを快適に提供することを目的とする。

SDI-1 の構成

- 喫茶店程度の広さをもつ SDI-1 設置場所

  不特定多数が出入りする。

- 通常の電話線

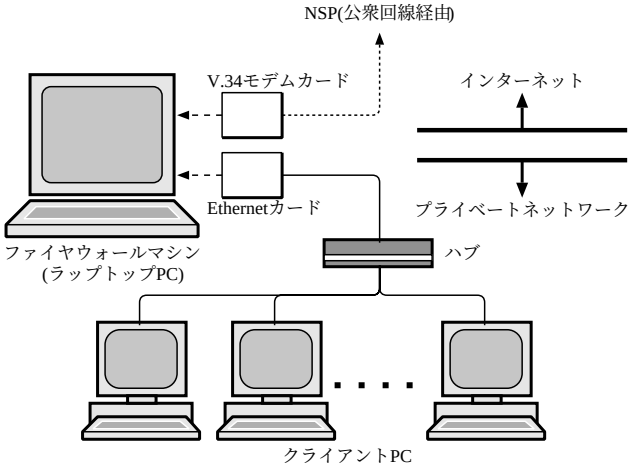

図 1.3: SDI–1 の構成

喫茶店などで日頃 FAX 用に使われている電話回線を転用するイメージ。

- ファイアウォール用ラップトップコンピュータ　　1 台

  PCMCIA V.34 モデムカード　　NSP 接続用
  PCMCIA Ethernet カード　　プライベートネットワーク接続用

- Windows95/Macintosh などのクライアント PC　　最大 8 台程度

  10Base-T Ethernet インタフェースを持っている。
  Netscape などのクライアントソフトウェアがインストールされている。

- 8 ポート 10Base-T ハブ　　数台

SDI-1 の接続形態

1. 10Base-T のハブで接続された数台の PC によるプライベートネットワークを構成する。
2. 1. はファイアウォールマシンに接続された V.34 モデム経由でインターネットに接続する (図 1.3)。

SDI-1 のサービス内容

プライベートネットワークユーザに対する WWW ブラウジング環境を提供する。

防火壁としての SDI-1

- インターネット側からプライベートネットワークへのログイン／アクセスは許さない。

↓

ファイアウォールマシンでは IP データグラムの転送機能を停止する。

- ファイアウォール上でアプリケーションゲートウェイを設定する。

WWW キャッシュとしての SDI-1

SDI-1 内のファイアウォール上で HTTP の Proxy サーバを動作させる。これによりプライベートネットワーク内部からインターネットへの WWW アクセスを可能とする。

SDI-1 を実際に運用する

SDI-1 モデルに沿った小規模ネットワークを実際に構築し、クラブイベントに集まった不特定多数のユーザに利用してもらった。

場所	西麻布 Club Yellow
イベント名	In dust real+Moon age
日時	1995 年 11 月 1 日 21 時 43 分
	〜1995 年 11 月 2 日 4 時 46 分
参加者人数	111 名
ネットワーク担当者	2 名
ネットワーク設営に要した時間	40 分
HTTP Proxy サーバ	CERN httpd
ファイアウォールマシン	TAXAN Power/V Note
	(486DX4 100MHz, 32M メモリ, 810M HDD)
	FreeBSD 2.1,
	V.34 モデムカード (XJ2288),
	Ethernet カード (3C589B)
クライアント PC	計 5 台
	PowerMac×3 台,
	PowerBook×1 台,
	Windows95×1 台

評価

HTTP Proxy サーバによるキャッシュの効果もあり、WWW ブラウジングのみを利用している分には実用的に運用することができた。

WWW アクセスの統計記録を表 1.3に挙げる。利用頻度は、一分に一度誰かが何かをクリックする程度と思われる。

SDI-2 への課題

今回の結果を踏まえ、今後以下の事柄を検討していきたい。

- SDI 構築支援キットの構成案‥より簡便なファイアウォール

表 1.3: WWW アクセスの統計記録

Number of HTML requests	350
Number of script requests	12
Number of non-HTML requests	1090
Number of malformed requests (all dates)	81
Total number of all requests/errors	1533
Average requests/hour: 217.6, requests/day	1533.0

  - WWW 用 Proxy サーバのインストール/設定/監視ツール
  - PPP 接続設定/監視ツール
  - ファイアウォールへのログイン/アクセス状況監視ツール
- WWW ブラウジング以外のサービスのサポート･･･限られた資源を活かす
  - IRC などのチャットサービス
  - WWW サーバによる各種情報提供サービス
  - CU-SeeMe などの画像を用いたサービス
  - RealAudio などの音声を用いたサービス

1.4 大規模な組織におけるプライベートアドレスへの移行と問題

大規模で組織内ネットワークの運用に歴史がある組織において、組織内のネットワークをプライベートアドレスを使用した体系に移行する際、いくつか懸念すべき点がある。

まず大規模な組織の場合には使用しているホスト数が多いため、IP アドレス自身の変更に掛かる純粋な作業工数が大変に大きいという問題があるのは自明である。しかしながらそれ以前に、以下のような大規模な組織に特有の問題がある場合が多い。

1. IP アドレス資源の枯渇問題に対する管理部門の無理解
2. 経路情報量の破綻状況に対する管理部門の無理解
3. 大規模なネットワークに対して円滑に移行作業を行なうための技術部門でのノウハウの欠如

特に 1. と 2. に関しては、管理部門においても知識として理解はしているが、経営上の判断から敢えて納得しないといった扱いがなされる場合がある。多くの場合そのような組織では、組織内のネットワークにおいて次のような状況が見られる。

- 運用上の切迫感がない。
- すでにポータブルな IP アドレスを取得している。
- 経路制御上の問題も生じていない。

事実上その組織においては、組織内ネットワークの運用に際して困っていないため、プライベートアドレスへの移行といったような作業は余計なコストを生じさせるだけであると判断してしまうことになる。これは私的な利益と公共の利益のどちらをどれだけ優先させるかという問題に帰着するため、簡単に答えを出すことはできないと思われる。

ここでは一つの事例を紹介する。A 社と、A 社の親会社である B 社との間で生じた問題である。この場合両社間には親子関係があるため、A 社は原則として B 社の意向に従わざるを得ないという事情がある。

当初 A 社と B 社は、それぞれ独自に異なるポータブルアドレスを取得して、それぞれの社内ネットワークをインターネットに接続して運用していた。ある時点において A 社と B 社の間にプライベートな IP リンクを設定することになり、両社の IP アドレスをどのようにすべきかが問題となった。

その時点においては既に RFC1597[82] が有効になっており、A 社内で一部プライベートアドレスの割り当てと利用も開始していた。A 社では最終的に、インターネット接続点にのみ ISP の割り当てる CIDR ブロックアドレスを使用し、A 社が独自に取得していたポータブルアドレスは IR に返却する予定で社内ネットワークのプライベートアドレスへの移行作業を進めていた。

ところが B 社との協議を進めて行く際に、B 社から A 社に対して次のような理由でプライベートアドレスの使用に関する異義が出された。

1. M&A を行なった場合、買収先がやはりプライベートアドレスを使っていた場合には IP アドレスの付け替えが必要となる。
2. ポリシーとしてポータブルアドレスを選択したい。

B 社は数万台以上のホストを有する大規模な社内ネットワークを運用しており、安定性を考えた場合に多少なりとも複雑になる可能性がある設定に関しては可能な限り避けたいという意志が働いたものと推測されるが、前述の大規模な組織に特有の問題が生じていた可能性もないわけではない。

1. については、プライベートアドレスの割り当て方法を工夫することによりある程度アドレスの衝突を防ぐことはできるが、100%の回避は不可能である。また 2. については純粋に考え方の問題なので反論して説得することは困難であろうと推測できる。広範囲に渡ってポータブルアドレスを使用して運用してきた組織では、敢えてプライベートアドレスに移行する必要はないという判断がなされるのも不自然ではない。

結局その異義に対しては、可能な限りポータブルアドレスを使うという方向で話を進めており、現在両社間においては、プライベートアドレスとポータブルアドレスの両方を使うという折衷案が検討されている。

第 2 章

ファイアウォールと非公式アドレスの運用

2.1 はじめに

ここでは、TIS Firewall Toolkit (FWTK) に代表されるデュアルホームのアプリケーションゲートウェイ型のファイアウォールの内部で、正式に取得したのではない非公式なアドレスを運用する場合の問題点について述べる。決して非公式アドレスの運用を推奨することが目的ではない。非公式なアドレスは、できるだけ速やかにプライベートアドレスあるいは正規のアドレスに移行すべきものであるが、現実には様々な理由により緊急にアドレスの移行が困難な場合もある。

2.2 非公式アドレスの問題点

デュアルホーム型のファイアウォールは一般に図 2.1の様に、外部セグメントと内部セグメントの両方に接続する位置に設置される。

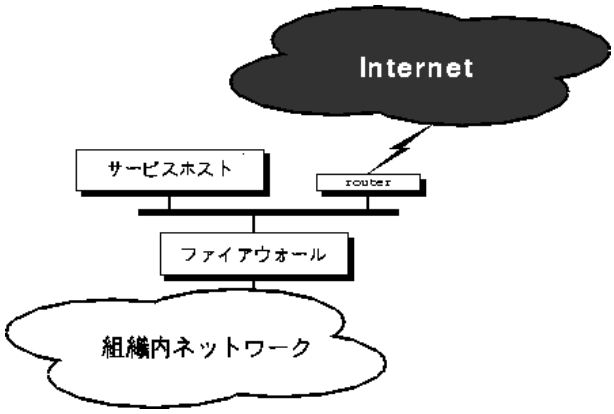

図 2.1: ファイアウォールの構成

ファイアウォールでは、IP の転送機能を無効にして、インターネット側のホストと組織

内ネットワークにあるホストが直接通信することをできなくする。その上でファイアウォール上でアプリケーション中継プログラムを動作させることで、内部のユーザがインターネットと通信することを可能にする。

この様な環境では、内部のホストはインターネット上のホストと直接通信する必要はない。そのため、内部のアドレスにはどのようなものを利用してもかまわないようにも感じられるが、実際にはそうではない。

前述の様にファイアウォールホストは、内部のホストとインターネット上のホストのどちらにも接続できる必要がある。ここで、内部ネットワークで非公式アドレスを運用していた場合、当然そのアドレスに対する経路情報は内側を向いていなければならない。

メールの送信などのために、ファイアウォールが内部と同じアドレスを正規に取得している組織と接続する必要が生じた場合、ファイアウォールホストはその組織のホストに対して SMTP によって接続しようとするが、そのアドレスに対する経路情報は内側を向いているので、パケットはインターネット側に出ていかない。また、相手から接続があった場合に、それに応答するパケットもまた内側に送られてしまうため接続は確立しない。したがって、この組織は、同じアドレスを使用している外部の組織とメールの交換ができないことになってしまう。メールに限らず、アドレスが衝突する組織とはあらゆる接続が不可能になってしまう。

このように、ファイアウォールによって組織内ネットワークがインターネットと遮断されている場合でも、内部で非公式アドレスを利用することによる問題は存在する。

2.3　非公式アドレスの運用の条件

以上のような問題を解決するために必要な技術的条件を考えてみる。ファイアウォールが接続することができない組織が外部にできてしまうのが問題であるから、その問題を解決すればよい。このためには、次の条件が必要とされる。

- ファイアウォールは内部で運用している非公式なアドレスに対する経路情報を持ってはならない

このためには、最低限次の条件を満たす必要がある。

- ファイアウォールの内側インタフェースのアドレスは非公式なものであってはならない

さらに、内側のホストに対する条件として次のものが挙げられる。

- ファイアウォールが通信する組織内のホストのアドレスは非公式なものであってはならない

ネットワーク	インターネット	バッファセグメント	非公式アドレス空間
ファイアウォール	○	○	×
サーバホスト	×	○	○
クライアント	×	○	○

表 2.1: ホストとネットワークの接続性

2.4 解決策

これらの条件を満たすためには、ファイアウォールの内側に正規アドレスあるいはプライベートアドレスのセグメントを設けて、ファイアウォールが通信する必要のあるサーバホスト等をそこに集める必要がある。このセグメントを**バッファセグメント**と呼ぶことにする。非公式アドレスを持つ内部のホストは、これらのサーバホストと通信できなければならない。

表 2.1に、ファイアウォール、サーバホスト、内部のクライアントが、それぞれどのようなネットワークと接続することができるかを示す。

このようにして、サーバホストを中継することで、クライアントはファイアウォールと直接通信することなく、インターネットのサービスを受けられるようになる (図 2.2参照)。図 2.2 に示した構成の要点を以下にまとめる。

- 外部にサービスを提供する WWW サーバ、DNS サーバは外部セグメントに置く。これらは同一計算機でも構わない。
- ファイアウォールの内側にはプライベートアドレス (または正式に取得したアドレス) を用いる。
- メールサーバ、ニュースサーバ、組織内用マスター WWW サーバはバッファセグメントに置く。
- バッファセグメントと残りの組織内ネットワークはルータによって接続される。
- バッファセグメントより内側のネットワークの運用は、インターネットとの接続には無関係である。ただし、この中にあるホストはすべてサーバマシンとの接続は可能であるが、非公式アドレスを使用しているホストはファイアウォールには接続できない。
- クライアントは、PROXY を指定してファイアウォールの機能を利用する。

次に、個々のアプリケーションがどのように利用されるかを考える。

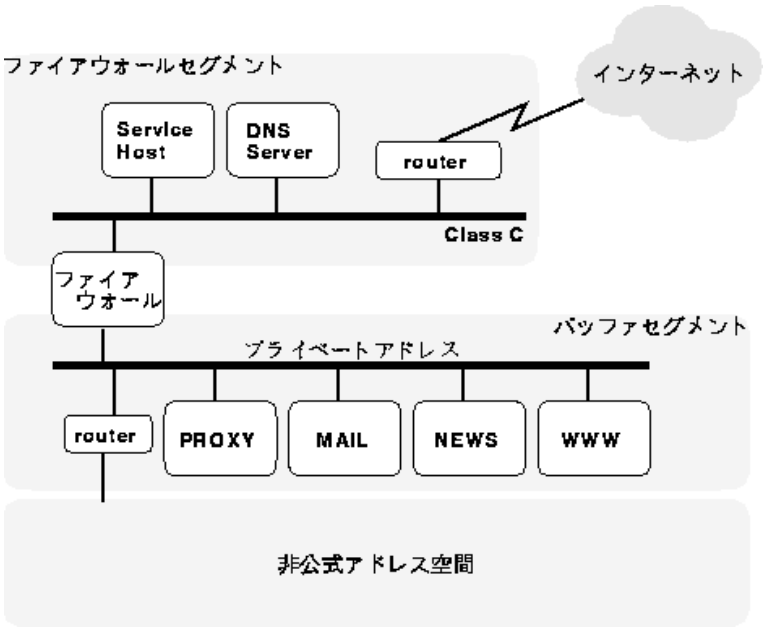

図 2.2: 非公式アドレスの運用例

電子メール

電子メールの運用は簡単である。ファイアウォールは組織内に対するメールはすべて内部のメールサーバに配送することができる。したがって、このメールサーバをバッファセグメント上に配置するだけで問題は解決する。メールサーバは組織内ネットワークとの接続性を持っているので、内部のクライアントや副メールサーバにメールを配送することができる。組織外にメールを発信する場合にはその逆に、このメールサーバに送るべきメールを集めればよい。

電子ニュース

電子ニュースの場合もメールとまったく同様のことが言える。内部のニュースサーバをバッファセグメント上に置くことで、クライアントはそのニュースサーバに接続することが可能であるし、組織内の他のニュースサーバにデータを送ることもできる。

WWW, Gopher

WWW や Gopher などの情報ブラウザの場合には、少し条件がある。一般にファイアウォールは、HTTP の proxy 機能を持っているが、クライアントが直接ファイアウォールに接続することはできない。

この場合、バッファセグメント上に HTTP の proxy サーバを設けて、クライアントはそのサーバからデータを受け取るようにする。組織内の proxy サーバがインターネット上のホストにアクセスする場合には、ファイアウォールの proxy 機能を利用

する。これは cascade proxy などと呼ばれる。

TELNET, RLOGIN

特別な手続きを使って telnet や rlogin のセッションを中継するアプリケーションゲートウェイの場合、同じ手続きをバッファセグメント上の proxy サーバと、ファイアウォール上で繰り返すことで、外部のサービスを利用することは可能である。ユーザはまず PROXY に接続して、そこでファイアウォールを指定して、さらに本当に接続したいサーバを指定することになる。この手続きは煩わしく、また PROXY からの接続は必ずファイアウォールに中継されるので、その処理を省くことができる。

telnet や rlogin は、単一の TCP セッションを使うプロトコルなので、内側の PROXY で、接続を無条件にファイアウォールに中継する処理を行えばよい。このためには FWTK に含まれる plug-gw や socks などのサーキットゲートウェイを利用することができる。PROXY に到着した接続は無条件にファイアウォールの telnet ポートに中継するので、ユーザは PROXY に接続したつもりが、いつのまにか自動的にファイアウォール上のアプリケーションゲートウェイと通信していることになる。これによって、ユーザに二段階の中継を行っているということを意識させない環境を作ることができる。

FTP

ftp の場合には自体は複雑になる。telnet や rlogin などと違い、 ftp はデータの転送に動的に新しい接続を作成するためである。しかもその接続は、ftp サーバ側からクライアントに対して作成される。ファイアウォールで動作するアプリケーションゲートウェイは、動的に生成される接続を自動的に中継する。したがって、PROXY ホストで ftp プロトコルを中継するために、サーキットゲートウェイを利用することはできない。

ftp プロトコルの自動的に特定のサーバに中継するためには、専用の中継プログラムが必要である。ただし、これは既存の ftp 用アプリケーションゲートウェイを若干修正することで比較的簡単に作成することができる。

図 2.2で、バッファセグメントと内部ネットワークをつなぐルータは、どれかのサーバに複数のネットワークインタフェースを用意して兼ねることができるし、すべてのサービスを同一ホストで提供することもできる。こうすれば、バッファセグメントにはホストが 1 台しかない単純な構成にすることも可能である。

意外と盲点なのは、バッファセグメントは必ずしもファイアウォールと接している必要はないという点である。非公式アドレスのネットワークを介して、プライベートアドレスや正規のアドレスを利用したネットワークが存在していても構わない。その場合、それらのネットワークからは、自由にファイアウォールと通信を行うことは可能なのである。地域的に分散する組織内ネットワークの場合、複数のサーバセグメントが存在することがあ

る。その場合には、分散するサーバセグメントに正式なアドレスを与えて運用することが可能である。

2.5　特殊な解決方法

今までに説明したのは、一般的な解決方法である。ネットワークに関して、ある特殊な状況を想定すれば、別の解決策を採ることも可能である。それについていくつか説明する。

衝突するアドレスが運用されていない場合　正式に割り当てられていないアドレスを内部で使用しているが、そのアドレスがまだどの組織にも割り当てられていない場合、または、割り当てられてはいるが、その組織がインターネットに接続していない場合には、衝突を意識せずに利用することができる。この場合、いつそのアドレスがインターネット側で有効なアドレスになるかわからないので、十分に注意して運用する必要がある。

衝突する組織がファイアウォールを運用している場合　衝突するアドレスを運用する組織がインターネット上に存在するが、その組織もファイアウォールを運用していて、限られたホストだけが到達可能な場合もある。この場合、それらの少数のホストに対するホスト経路をインターネット側に向けることで、ほとんどの場合支障なく運用することが可能である。もちろん、その少数のホストと同じアドレスを持つ内部のホストはファイアウォールと通信することはできない。

この場合も、相手の組織がいつホストのアドレスを変更したり、増やしたりするかはわからないので、十分な注意が必要である。

諦める　衝突するアドレスを運用している組織とは絶対に通信の必要はないと仮定して、それを無視するという方法も考えられる。この場合、自分が相手に通信できないだけではなく、相手から自分の組織にメールを送るような場合にも問題が生じることを忘れてはならない。これは最悪の方法なので、短期間に限っての暫定的な運用でなければ、絶対に選択すべきではない。

2.6　おわりに

非公式なアドレスを運用せざるを得ない状況で、どのような方法が考えられるかについて述べたが、これはあくまでも一時的な対策である。定常的な運用形態として採用するのではなく、正式なアドレスに移行するまでの暫定的な運用でなければならない。

第 3 章

SSL wrapper

3.1 SSL wrapper

インタネット経由で通信するアプリケーションに暗号通信機能を提供するプロトコルとして Secure Sockets Layer (SSL)[66] が提案されている．また，これを元に作成された実装もいくつか発表・配布されている．これらの実装を用いることにより従来のアプリケーションで暗号通信が可能となる．しかし，このためにはサーバプログラム及びクライアントプログラムについて若干の変更作業，再コンパイル作業が発生する．

そこで我々は変更作業を極力少なくして暗号通信を実現するための通信方式を提案し考察を行ったので報告する．

3.2 はじめに

近年，インタネットの拡大に伴ってインタネットを利用した様々なサービスが普及している．しかし，利用者が増えるにしたがってネットワークの盗聴によるパスワードの詐取やなりすまし等による不正アクセス，不正侵入事件等が発生し問題となっている．このような問題を解決する手段の一つとして通信内容の暗号化および通信相手の認証があげられる．

現在，提案されている暗号通信 (認証) プロトコルとして

1. IPv6 のセキュリティオプション [87]
2. IP/Secure[88]
3. Secure Sockets Layer(SSL)
4. S-HTTP[67]
5. PEM[89, 90, 91, 92]
6. PET[93]

などが提案されている．またいくつかサンプル実装も発表されている．それぞれ S-HTTP や PEM, PET はアプリケーション層，SSL はトランスポート層とアプリケーション層の中間，IP/Secure と IPv6 はネットワーク層に位置している．

ここで実際にこれらのプロトコルを実装する場合に S-HTTP のようなアプリケーション層に変更を加える場合は汎用性が無くなり，IPv6 のようなネットワーク層での変更はトランスポート層以上に対する影響が大きくなることが考えられる．

このような条件で我々は SSL に注目し，各アプリケーションについて SSL を用いて暗号通信を実現することにした．しかし SSL を用いる場合でも API[1]にしたがってサーバプログラム，クライアントプログラムの両方を修正・再コンパイルする必要が生じる．そのためベンダが提供しているようなバイナリでしか配布されていないコマンド，サーバ等は SSL を用いて暗号化することができない．

そこで本稿ではまず，SSL の概要を説明し，次に SSL の一実装である SSLeay パッケージについて説明する．そして今回提案する通信方式について述べ，最後に考察を行う．

3.3　SSL

3.3.1　概要

SSL はサーバ・クライアント間の通信において盗聴防止や改ざん防止，認証機能を提供するプロトコルである．SSL は二つの層から構成されている (図 3.1)．

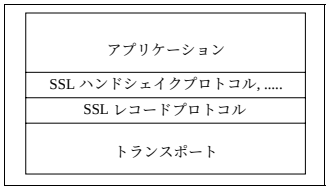

図 3.1: SSL の構成

そのうち下位層は TCP のような信頼性のあるトランスポート層の上に位置する SSL レコードプロトコルである．このプロトコルは上位のプロトコル群のカプセル化を担当している．このようにカプセル化されるプロトコルの一つに SSL ハンドシェイクプロトコルがある．これはサーバ・クライアント間の認証や暗号に関するパラメータの交換を行う．SSL は次の性質を持つように通信路のプライバシを提供する．

- 通信内容は DES[94] や RC4 のような対象暗号アルゴリズムで暗号化され当事者以外には分からない

[1]実装により異なる

- 通信相手は RSA[95] のような非対象暗号アルゴリズムで認証される
- 通信内容は MD5[96] のようなハッシュ関数で署名され途中で変更されても分かるように完全性を持つ

3.3.2 プロトコル

SSL ではクライアントがサーバに接続する際に SSL ハンドシェイクプロトコルを用いて以下の情報の交換を行い，両者の間で暗号アルゴリズムやそのパラメータの設定および認証を行う (図 3.2).

- プロトコルバージョン
- サポートしている暗号アルゴリズム
- 証明書
- セッション鍵

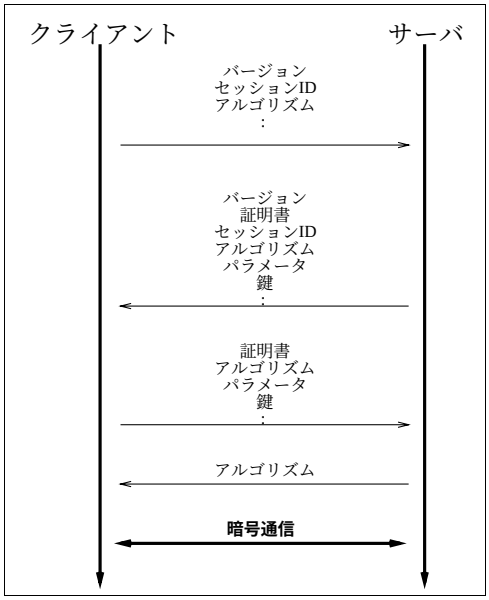

図 3.2: SSL ハンドシェイクプロトコルの一例

上記のプロトコルで両者間のアルゴリズム，パラメータが決定すると，それにしたがって暗号通信が行われる.

3.3.3 SSLeay

SSL プロトコルを元にいくつかの実装が公開されている.

SSLREF[97] Netscape 社の実装したパッケージ．アメリカ国外には持ち出すことができない

SSLeay[98] オーストラリアの Eric Young 氏が開発したパッケージ

SSLPref[99] イギリスの Jeremy Brandley 氏が中心に活動しているプロジェクト (まだ実装途中である)

SSL パッケージの特徴は以下の通り:

- SSL バージョン 2 を実装
- 以下のプラットホームで動作
  - SunOS 4.1.3
  - SunOS 5.x
  - HP-UX 9.x
  - IRIX 5.x, 4.x
  - DGUX 5.x
  - OSF 1.x
  - AIX 3.2
  - BSD/OS 2.0.1
  - (Windows 3.1 ; まだ半分)
- 以下のアプリケーションで動作
  - telnet
  - ftp
  - NCSA Mosaic 2.5, 2.6b1 2.7b2
  - NCSA httpd 1.3, 1.4.2
- 利用できる暗号は以下の通り
  - RC4
  - IDEA

  - DES
  - RSA

- 証明書は X509 の形式
- 全て Eric 氏が作成した

3.3.4 API

SSLeay で定義されている主な API を以下に簡単に述べ，最後にクライアントとサーバのプログラム例を述べる．
SSL **コンテキスト**，SSL **ハンドル** SSL のコネクションに関する情報を SSL コンテキストと呼ぶ．また SSL のコネクションの管理用データ構造を SSL ハンドルと呼ぶ．複数のハンドル (= コネクション) で一つのコンテキストを共有しても構わない．コンテキストで保持している主な情報は以下の通り:

- 証明書
- 暗号アルゴリズム
- 鍵
- コネクション ID

コンテキスト，ハンドルに関連する API は表 3.1の通り．

`SSL_CTX *SSL_CTX_new(void)`	コンテキストの生成
`SSL *SSL_new(SSL_CTX *ssl_ctx)`	ハンドルの生成
`void SSL_free(SSL *s)`	ハンドルの解放

表 3.1: コンテキスト関連 API

秘密鍵

表 3.2の API(のどれかを利用して) で SSL に対して秘密鍵を指定する．

証明書の登録

表 3.3の API(のどれかを利用して) で SSL に対して証明書を指定する．サーバは**必ず**証明書を持たなくてはいけない．クライアントはどちらでも良い．

なお SSLeay では次の二種類のフォーマットを扱うことができ SSLeay では PEM を推奨している．

`int SSL_use_RSAPrivateKey(SSL *s, RSA *key)`	秘密鍵の指定
`int SSL_use_RSAPrivateKey_file(SSL *s, char *file, int type)`	秘密鍵の指定 (ファイル名による指定)
`int SSL_use_RSAPrivateKey_fp(SSL *s, FILE *fp, int type)`	秘密鍵の指定 (FILE 構造体による指定)

表 3.2: 秘密鍵指定用 API

`int SSL_use_certificate(SSL *s, X509 *cert)`	証明書の登録
`int SSL_use_certificate_file (SSL *s, char *file, int type)`	証明書の登録 (ファイル名による指定)
`int SSL_use_certificate_fp(SSL *s, FILE *fp, int type)`	証明書の登録 (FILE 構造体による指定)

表 3.3: 証明書の登録用 API

- DER
- PEM(base64)

通信部分

SSLeay ではトランスポート層による通信路はあらかじめユーザが socket(2) などを使って相手と接続しておく．サーバ側，クライアント側のどちらも SSL_new() で SSL ハンドルを作っておき，SS_set_fd() と言う API で通信チャネルと関係付けを行う．一度ハンドルとの関係付けが行われた後は，このハンドルを用いて通信を行う (表 3.4).

プログラム例

以上の API を用いたクライアント，サーバ両方のプログラム構成例を挙げる．
クライアント側

```
/* SSL コンテキストを生成する */
ctx = SSL_CTX_new();

/* SSL ハンドルを生成する */
con = SSL_new(ctx);

/*
 * 通常の socket(), [bind()], connect() を呼び，サーバと接続する
 * s : file descriptor
```

void SSL_set_fd(SSL *s, int fd)	SSL ハンドルとファイルディスクリプタとの関連つける
int SSL_connect(SSL *s)	サーバへの接続要求 (クライアント用)
int SSL_accept(SSL *s)	クライアントから接続受付 (サーバ用)
int SSL_read(SSL *s, char *buf, int len)	データの受信
int SSL_write(SSL *s, char *buf, intlen)	データの送信

表 3.4: 通信部分の API

```
 */

/* その file descriptor と con を結びつける */
SSL_set_fd(con, s);

/* SSL でのコネクションを確立する */
SSL_connect(con);

/*
 * SSL_read, SSL_write で読み書きを行う
 */
           :
```

サーバ側

```
/* SSL コンテキストを生成する */
ctx = SSL_CTX_new();

/* SSL ハンドルを生成する */
con = SSL_new(ctx);

/*
 * 通常の socket(), bind(), listen(), accept() を呼びクライアントと接続する
 * s : file descriptor
 */

/* 秘密鍵を指定する */
SSL_use_RSAPrivateKey_file(con, "server.rsa");
```

```
/* 証明書を指定する */
SSL_use_cerificate_file(con, "server.cert");

/* クライアントと暗号アルゴリズムなどの調整を行う */
SSL_accept(con);

/*
 * SSL_read, SSL_write で読み書きを行う
 */
          :
```

3.4　設計

上で述べたように SSLeay を使ってサーバ，クライアントの両方を修正しコンパイルすれば SSL が利用可能となる．しかし，この場合サーバ，クライアントのソースが必要であり，ベンダ提供のコマンドのようにソースコードが無い場合に対応ができなくなってしまう．また，ソースコードがある場合でも全てのコマンドについて変更を行い，再コンパイルを行う必要が生じる．

そこで，これらの変更作業を少なくするためにクライアント・サーバ間に SSL を用いた平文 → 暗号，暗号 →平文の変換を行うプロセスを導入した以下の通信方式 (図 3.3) を提案する．提案方式の概要は以下の通り:

- ほとんどのサーバが inetd 経由で起動されることに注目し，暗号/復号を行う中継プロセス (sslwrapper) を導入して SSL を利用可能にする
- クライアント側は socks[100] のプロトコルを拡張し UNIX のプロセス間通信の API を置き換える．クライアントは，この置き換えた API にしたがって上記の sslwrapper に相当する中継プロセス (sslrized sockd) と通信し，これがサーバと通信する

本方式の利点は以下の通り:

- サーバ側のコードを変更する必要が無い
- クライアント側もコードの修正無く再コンパイル (リンク) だけですむ．また，ダイナミックリンクや共有ライブラリを利用している場合には，そのライブラリを入れ換えれば良いため，コンパイルの必要が無い

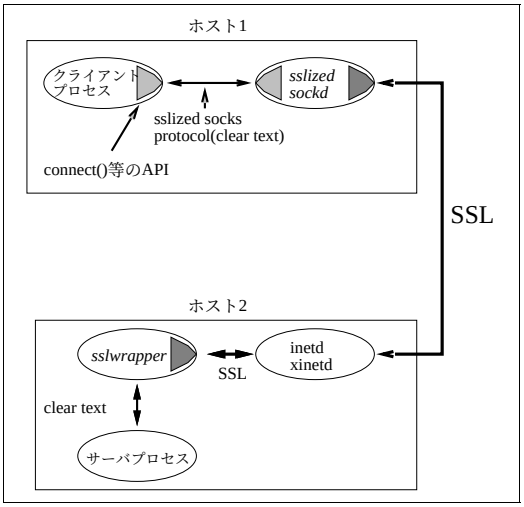

図 3.3: sslwrapper と sslized sockd

3.4.1 sslwrapper

sslwrapper は inetd から起動され，コマンドラインに指定されたサーバプログラムを起動する．サーバプログラムとの通信はソケットを用いる (あらかじめ接続しておく)．標準入出力に対して SSL_accept() を実行し，外部との通信は SSL を用い，サーバプログラムの中継を行う．また sslwrapper はコマンドラインでサーバプログラムの他にサーバの証明書を指定する必要がある．

3.4.2 sslrized sockd

クライアント側の中継プロセスである sslrized sockd は拡張した socks プロトコルでクライアント側から要求を受け，サーバ側へ SSL_connect() で接続を要求したあと両者の中継を行う．

socks プロトコルを拡張するのは SSL での通信を希望するクライアントと希望しないクライアントの両方を扱うためである．そのため socks のコマンド部分に通常の接続を要求する CONNECT コマンドの他に SSL での接続を要求する SSLCONNECT コマンドを追加する (表 3.5)．

なおクライアントと sslrized sockd との間の通信は平文で流れるため，盗聴を防ぐために両者は同一ホストで動作する必要がある．または，クライアントと sslrized sockd の間の通信を暗号化すれば良いが，これは意味が無い．

コマンド	動作
CONNECT	sockd に対してサーバへの接続を要求する
BIND	sockd に対して他サーバからの接続要求の受け付けを要求する
SSLCONNECT(追加)	sockd に対して SSL でのサーバへの接続を要求する

表 3.5: socks プロトコルのコマンド

3.5　考察

今回提案した方式について以下の項目に注目して考察を行なった．

認証 本方式では通信内容の暗号化のみを行うだけでサーバ，クライアントの認証は行っていない．認証機能を実現するにはクライアント側ではクライアント・sockd 間での，サーバ側では inetd ・サーバ間での認証が必要となる．socks version 5[101] ではクライアント・sockd 間の認証機能が追加されているので，この機能を利用する方法も考えられる．

他の暗号プロトコル 現在は SSL しか利用できないが，今後 GSS-API[102, 103] のようなプロトコル，API が登場することは十分考えられる．

ftpd 問題 ftpd は inetd から起動されるが，データ転送時等に別の通信路を作る (サーバからクライアントへの方向とクライアントからサーバへの方向の両方向)．そのため，新たに接続された通信路では暗号化を行うことができない．そこでサーバプロセスの動作しているホスト上でも sslrized sockd を動作させ，更に socks プロトコルを拡張し BIND コマンドを拡張して SSL による接続要求の受け付けを行う SSLBIND を追加することによって暗号化通信を行うことは可能になる (図 3.4)．しかし，この解決方法では

- サーバ側でも修正作業，再コンパイルが発生してしまう
- sslwrapper 経由の通信路と別の SSL コンテキストを使うことになる
- (美しくない?)

という問題が生じる．

3.6　まとめ

本稿では SSL プロトコルを用いたサーバ・クライアント間の暗号通信を少ない変更で導入可能な方式を提案した．本方式ではサーバプログラムの大部分が inetd 経由で起動される

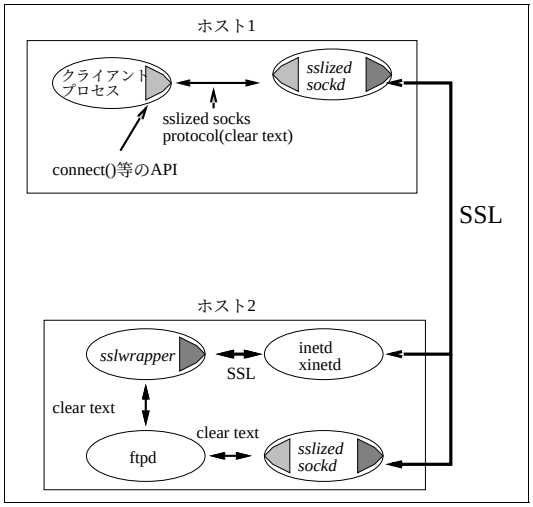

図 3.4: ftpd 問題

ことに注目してサーバと平文で，クライアントと SSL で通信する中継プロセス sslwrapper を導入することでサーバコードの変更作業を無くした．またクライアントも socks のプロトコルを拡張し SSL での接続を要求するコマンドを追加することにより再コンパイルだけで利用可能にした．

また本方式についての考察を行った．今後の課題を以下に示す．

- 認証機能
- SSL 以外の暗号プロトコルの利用
- 新たに通信路を生成するサーバ/プロトコルへの対応

第 4 章

おわりに

IP アドレス不足とインターネットバックボーンの経路制御表の爆発に対する短期的解決としてのプライベートアドレスをファイアウォールの内部で使用することについて、プライベートアドレスではなく非公式アドレスをファイアウォールの内部で使用することについて、既存通信プログラムが SSL を使えるようにするための方法とについて報告した。

来年度は管理 Tool, ファイアウォールモデル、ファイアウォール以外からの抜け穴などの新しいファイウォールについていろいろな角度から研究する。