Setting Up Your New Domain Mini−HOWTO.

Table of Contents

Table of Contents

Setting Up Your New Domain Mini−HOWTO.

by Christopher Neufeld (neufeld@physics.utoronto.ca) (neufeld@thepuffingroup.com).

version 0.5. 1999/12/23.

This document outlines the things you will probably have to do when you want to set up a network of computers under your own domain. It covers configuration of network parameters, network services, and security settings.

6.Deciding Which Domain Services You Will Host

7.Configuring Your Hosted Services

8.Securing Your Domain

9.Acknowledgements

10.Glossary of Terms

1.Notices

1.1 Disclaimer

This is a very preliminary document. I have glossed over many things which could be given in much more detail, and have probably missed important sections entirely. I need feedback on this document before I can make it widely available. Any suggestions for additions, deletions, or areas where I ought to provide more or less detail are very welcome.

1.2 Copyright

2.Introduction

This is a guide to setting up your own domain of Linux machines, or mixed Linux and Windows machines, on an always−up connection with a static IP and a named domain. It is not really intended for setups which use dynamic IPs, or which are regularly disconnected from their provider for long periods of time, though some basic hints for operating such a setup are available in section Using A Dynamic IP.

With the increasing availability of permanent connections and static IPs, it's becoming easier for people and organizations to set up a real domain, with the associated Internet presence. Proper planning at the outset can reduce problems later.

Much of this document describes techniques for implementing unobtrusive security on the newly exposed network. This deals with protection from external attack, and from casual internal attack. It does not claim to provide an extremely secure setup, but is usually enough to discourage the less determined attacker.

This document is primarily directed at small organizations which have an existing network of computers, possibly with a shared dialup line, which are trying to move to a permanent, relatively high−speed connection, either to improve data transfer with the outside world, or to create a WWW or FTP site. The document is also directed at new organizations which want to skip the early stage and start out with higher speed networking and services under their own domain name.

Throughout this document, I will discuss the configuration of a newly registered domain, **example.com**. Note that the name example.com is reserved by the Internet Assigned Numbers Authority for use in documentation, and so will never correspond to an actual domain.

Much of the information in this document is available in other places. I have tried to distill the material relevant to the creation of a new domain. Where detail on a specific subject is lacking, you may want to consult one of the more comprehensive documents.

This document will also assume a mixed OS environment. Specifically, I will assume that some desktop machines are running some version of Microsoft Windows, while servers and the private network gateway are running Linux.

3.Planning Your Network Topology

While there are arguments which can be made for many different network layouts, the requirements of many organizations can be met by putting the desktop machines and private servers on a private masqueraded subnet, and the publicly accessible machines on valid external IPs. The machines on valid external IPs will be referred to in this document as ``exposed hosts''. This leads to the following (example) topology:

```
+--------------+
|              |               +---------------+
| ISP-supplied |---------------| FTP server    |
| router       |        |      +---------------+
|              |        |
+--------------+        |      +---------------+
                        |------| WWW server #1 |
                        |      +---------------+
                        |
                        |      +---------------+
                        |------| WWW server #2 |
                        |      +---------------+
                        |
                        ~
                        ~
                        |
                        |      +---------------+
                        |------| Private       |
                               | Network       |
                               | Gateway       |
                               +---------------+
                                      |
                                      |
                                      |
                                      |
     +------------+                   |      +-------------------+
     | Desktop #1 |-------------------|------| Private server #1 |
     +------------+                   |      +-------------------+
                                      |
            .      -------------------|--------        .
            .                         |                .
            .      -------------------|--------        .
                                      |
     +------------+                   |      +-------------------+
     | Desktop #N |-------------------|------| Private server #N |
     +------------+                          +-------------------+
```

In this example, the router provided by the ISP (Internet Service Provider), FTP server, WWW servers, and the machine labelled ``private network gateway'' all have externally visible IP numbers, while the desktop and private server machines have IP numbers allocated from RFC 1918, reserved for private use. The IP numbers you choose for use within the private network (everything below the private network gateway machine) should be chosen to be unique, not only among the hosts under your control, but should also not conflict with numbers assigned on similar private subnets at other sites or partner companies with whom you might, at some time, want to implement a virtual private network, in order to reduce confusion and reconfiguration when the networks are merged in that way. As outlined in the RFC, you can choose from any class C network from 192.168.0.* to 192.168.255.*, or any class B network from 172.16.*.* to 172.31.*.*, or the class A network 10.*.*.*. In the rest of this document I will assume that your private network (if you've chosen to create one) is on the class C network 192.168.1.*, and your private network gateway machine is at IP number 10.1.1.9, one of the IP numbers provided to you by your provider (note that this is not a valid external IP, I use it as an example only). I will also assume that there is a machine, betty.example.com, at 10.1.1.10, which will handle both www and FTP services.

Take note of the number of external IP numbers which you need for your own machines. You will need one IP number for each machine which lies outside the private network gateway, plus one for the gateway itself. This count does not include any IP numbers which may be taken by routers, broadcast addresses, and so on. You should ask your provider for a block of addresses large enough to mount the given number of machines. For example, in my office network, of the 8 IP numbers allocated from the ISP, three were not usable by my computers, leaving enough IP numbers for four machines outside the gateway, plus the gateway itself.

This network topology is not correct for everybody, but it is a reasonable starting point for many configurations which don't have special needs. The advantages of this configuration include:

- Easy expandability. If you suddenly double your number of private nodes, you don't have to worry about getting a new IP block from your provider and reconfiguring all of the interfaces on your machines.
- Local network control. Adding a new workstation to your private network requires no communication with your provider, unlike exposed nodes, which need both forward and reverse DNS (domain name service) mappings if they are to perform certain tasks (ssh and ftpd may complain if they can't perform reverse and forward DNS on incoming connections). A reverse DNS query is an attempt to obtain the host name from the IP number.
- Centralized security. The private network gateway can enforce security over the whole private network, filtering packets and logging attacks, rather than having to install such measures on each desktop and server on the private network. This can be enforced not only on incoming packets, but also on outgoing packets, so that a misconfigured desktop machine doesn't inadvertently broadcast data to the outside world which ought to remain internal.
- Easy transplantability. Because the IP numbers within the private network are yours for as long as you want them, you can move the entire network to a new range of IP numbers without having to make any changes to the network configuration on the private network. The publicly exposed hosts still have to be reconfigured, of course.
- Transparent Internet access. The machines on your private network can still use FTP, telnet, WWW, and other services with minimal obstruction, assuming a Linux masquerading router. The users may not even be aware that their machines are not on externally visible IP numbers.

Some of the potential disadvantages of such a configuration are:

- Some services will not be available directly to the machines on the internal network. NTP synchronization against an outside host, certain obscure services which may not have masquerading rules in the kernel, and .shosts authentication for logging in to external nodes are all difficult or impossible, but simple workarounds are almost always available.
- More network hardware costs. The private network gateway machine needs two network cards, and you need at least two hubs / switches, one on the visible network and one on the private network.

- Machines outside the private network cannot easily make direct connections to machines within the private network. They may have to open a session first on the private network gateway machine, then log through to the internal host. It is possible to route packets transparently through the firewall, but this is not recommended for security reasons which will be discussed in a later section.

You should consider these points in planning your network topology, and decide if a fully visible network is more appropriate for your situation. In the rest of this document I will assume that you have configured your network as shown above. If you have chosen to have a fully visible network, some details will differ, and I will try to point out such differences in this document.

As a special case, if you do not need any external servers, the ISP−supplied router can be attached directly to your external interface on the private network gateway machine, rather than with a hub.

4.Obtaining Your Connection

4.1 Choosing Your Provider

As with anything, shop around. Determine which services are available in your area, as well as the costs associated with those services. Not all locations are wired to accept DSL, and some locations may not be suitable for wireless connections due to constraints of the landscape, architecture, or environment. Be prepared to provide the street address of the location where your hookup will be installed, as DSL speeds are strongly dependent on your distance from the switch, and ask specifically about such details as bandwidth between your machine and the provider, what has to be done to install the connection, and what hardware is provided in the quoted monthly rate. Also, you should have some idea of how many IP numbers you need for your own machines (remember that not all IP numbers in the block you get from the provider will be available for attaching your computers). Ask the provider what their total bandwidth is out to the outside world, as the quoted speed is only between your site and theirs. If the provider has insufficient bandwidth to the outside, the customers will suffer bottlenecks within the provider's network.

Once you have narrowed down a list of candidates, ask around, see if anybody can provide you with recommendations for the services you're considering. Ask them what sort of bandwidth they get to unloaded sites. Also, if you intend to have fast connections between the new domain and local ISP accounts from home, for telecommuting, or just remote administration, it is essential that you do a *traceroute* from your home ISP account to a host operating on the service you're considering. This will tell you how many hops, and how much latency you should expect, between home and the new domain. Latencies much above 100 to 200 milliseconds can be difficult to use for extended periods of time. The *traceroute* should be run around the time of day that you expect to make use of the network connection between home and the new domain.

4.2 Preparing For Hardware Installation

After you have chosen the provider and service type for the new domain, ask about installation details. You may require service calls from the telephone company as well as from the ISP in order to install the service, and the technicians may need access to controlled areas of your building, so inform the building engineer of the installation requirements.

Before the ISP technician arrives, ask for the network parameters, specifically the IP number, netmask, broadcast address, gateway routing address, DNS server address, and also what cabling you need to connect to the hardware delivered by the technician (i.e. straight−through or crossover RJ45 cabling, etc.).

Have one machine available for testing, and put it close to where the network connection hardware will be installed. If possible, configure it before the service technician arrives, setting the IP number and netmask, and have the appropriate cabling ready so that the installation and testing can be done quickly.

4.3 Testing The Connection

With your test machine attached to the ISP's hardware, make sure that you can ping sites beyond the ISP. If not, a *traceroute* to the outside can help to show where the connection is failing. If traceroute shows no successful hops it indicates that your test machine's network configuration (default route, interface address, NIC drivers, DNS, etc.) is incorrectly set. If it shows one hop, that could mean that your router is not correctly configured to communicate with the ISP. If it shows several hops before failing, the problem is almost certainly in the ISP or in the outside world, and beyond your immediate control.

4.4 Using A Dynamic IP

The benefits of a corporate connection, with a static IP block and various hosted services, comes with a cost. It can be more than ten times as expensive as a high speed home connection on DSL or cable modem. If the budget can't support a corporate connection, or if no such connections are available in your area, you might want to try to set up a domain on a dynamic IP. Instead of a range of IP numbers, you typically get exactly one, which means that your private network gateway machine will also have to host any incoming services from the outside.

First, you might want to check the legality of it. Many companies' user agreements explicitly forbid setting up externally−accessible servers on personal accounts. They may enforce this with packet filters blocking incoming connections on the http and FTP ports. You should also be aware that the quoted connection speed for personal accounts such as home DSL or cable modem are the downlink speeds, and that the uplink speeds might be much slower. The uplink speed is what is important for serving up FTP or web content.

If you have a dynamic IP, and you want to have incoming connections, you will have to subscribe to a dynamic IP hosting service, such as DynIP, DynHOST, or TZO. These all work by running software on your machine which passes your current IP number on to the company's servers. When your current IP number arrives at the servers, their DNS tables are updated to reflect the new value. You can either get a domain name under their domain name, such as ``example.dynip.com'' or ``example.dynhost.com'', or you can register your own domain and set the primary DNS authority to point to the company providing this service (usually at a higher cost).

There is also a free hosting service, at Domain Host Services. They seem fairly new, and there are few details on their web site at the moment, but you might find it worth a look.

If you have set up a dynamic IP, and subscribed to one of these services, it will affect some of the decisions you make in section Deciding Which Domain Services You Will Host. In particular, there is little point subscribing to a dynamic IP hosting service if you do not plan to host at least one of web or FTP services. You will have to set primary DNS authority to point to the company you've chosen. You should not have a *named* daemon answering requests from outside your private network. Other details, such as handling of email, will depend on the specifics of the service you've subscribed to, and can best be answered by the support staff of that company.

One final note: if you want to have remote access to a machine with a dynamic IP, but don't need it for hosting other services, the inexpensive solution is to create a ``drop box'' on a publicly accessible machine with a static IP, and have your dynamic IP host send its IP number there, either in email or simply by writing it into a file on a shell account. When you want to access your machine remotely, first extract the current IP number from the drop box, then use *slogin* to attach directly to that IP number. This is, after all, really all that a dynamic IP hosting service does, they just do it automatically over standard services, saving you some steps.

5.Registering A Domain Name

In order for people in the outside world to locate your servers under the domain name of your choice, whether for web, FTP, or email delivery, you will have to register the domain name for insertion into the relevant top level domain database.

Exercise some simple prudence in choosing your domain name. Certain words or phrases may be forbidden on the grounds of community standards, or may be offensive to visitors whose language or slang differs from that of your region. Domain names can contain only the 26 letters of the Roman alphabet (without accents), the hyphen (though not at the beginning or end of the name), and the 10 digits. Domain names are not case−sensitive, and can be at least 26 characters long (this limit is subject to change). Be careful not to register a name which you can reasonably have been expected to know infringes on the trademarks of an existing company, the courts are not kind to cybersquatters. Some information on the circumstances under which your poorly−chosen domain name might be stripped from your control are available in this Uniform Domain Name Dispute Resolution Policy.

There are many companies which register names in the ``.com'', ``.net'', and ``.org'' top level domains. For a current list, check the list of accredited registrars.

To register a name under a country top level domain, such as a ``.ca'', ``.de'', ``.uk'', etc., check with the appropriate authority, which can be located in the Country Code Top−Level Domains database.

Typically, you have to provide the registrar with contact information, primary and secondary DNS IP numbers, a change request validation scheme (you wouldn't want just anybody changing your domain for you), and money in the form of an annual fee. If you're not comfortable with the change request validation schemes offered by a registrar, let them know that you're not willing to use the service until they address your security concerns.

6.Deciding Which Domain Services You Will Host

Most full−service ISPs will provide a variety of domain services for their customers. This is largely because of the problems associated with hosting these services under certain other, more popular desktop and server operating systems. These services are much easier to provide under Linux, and can be hosted on fairly inexpensive hardware, so you should decide what services you want to take on for yourself. Some of these services include:

- Primary DNS authority on your domain. See section Primary DNS Authority.
- Electronic mail. See section Electronic Mail.
- Web space hosting. See section Web Space Hosting.
- FTP space hosting. See section FTP Site Hosting.
- Packet filtering. See section Packet Filtering.

In each of these, you basically have to weigh convenience against control. When your ISP performs one or more of these services, you can usually be fairly sure that they have people with experience maintaining the service, so you have less to learn, and less to worry about. At the same time, you lose control over these services. Any changes require that you go through the technical support of your ISP, something which may sometimes be inconvenient or cause longer delays than you would like. There's also a security issue involved, the ISP is a much more tempting target to attackers than your own site. Since an ISP's servers might host email and/or web space for the dozens of companies which are their customers, an attacker who compromises one of those servers gets a much higher return for his efforts than one who attacks your personal servers, where only one company's data is kept.

6.1 Primary DNS Authority

When a person somewhere in the outside world attempts to connect to a machine in the new example.com domain, queries are sent between various servers on the Internet, ultimately resulting in the IP number of that machine being returned to the software of the person attempting the connection. The details of this sequence are beyond the scope of this document. Neglecting many details, when a request is made for the machine fred.example.com, a centralized database is consulted to determine what is the IP number of the machine which holds primary DNS authority for the example.com domain. This IP number is then queried for the IP number of the machine fred.example.com.

There must be a primary and a secondary DNS server for every domain name. The names and IP numbers of these two servers are stored in a centralized database whose entries are controlled by domain registration authorities such as Network Solutions.

If you elect to have primary DNS authority hosted by your ISP, these two servers will probably both be machines controlled by the ISP. Any time you want to add an externally visible machine to your network, you will have to contact the ISP and ask them to put the new machine in their database.

If you elect to hold primary DNS authority on your own host, you will still use another machine as your secondary. Technically, you should use one on a redundant Internet connection, but it is very common that the secondary is held on one of your ISP's machines. If you want to add an externally visible machine to your network, you will have to update your own database, and then wait for the change to propagate (something which takes, typically, a small number of hours). This allows you to add barney.example.com without having to go through your ISP.

It is a good idea to set up secondary DNS on a geographically distant host, so that a single cable cut near your ISP doesn't take both your primary and secondary DNS servers off line. The domain registrar you used to register your domain name may provide secondary DNS service. There is also a free service, Granite Canyon, available to anybody who asks.

Regardless of whether or not you choose to act as primary DNS authority for your domain, see section Setting Up Name Resolution for configuration help. You will want some sort of name resolution system for your private network, even if you delegate primary DNS authority to the ISP.

6.2 Electronic Mail

When you subscribe with your ISP, they will typically supply a number of email boxes. You can elect to use this service exclusively, in which case all incoming email is stored on the ISP's servers and your users read their mail with POP3 clients which connect to the ISP's servers. Alternately, you may decide to set up email on your own machines. Once again, you should weigh the merits of the two approaches, and choose the one which you prefer.

Things to remember if you use the ISP for all email:

- It may be easier to access the email from home, or from other locations when you're on a business trip, depending on the security which you use to protect your domain.
- Email is routinely stored on the ISP's servers, which may be a problem if sensitive material is sent

unencrypted.
- You have a limited number of email accounts, and may have to pay if you exceed this limit.
- To create a new email address, you have to go through the ISP.

Things to remember if you provide your own email:

- Email is routinely stored on your own servers, with backup storage on your ISP if your mail host goes down or its disk fills up.
- You have an essentially unlimited number of email accounts, which you can create and delete yourself.
- You have to support the email clients used on your private network, and possibly by people trying to read their email from home.

One possible approach is to host email yourself, but also use the several email addresses provided by the ISP. People who need email accessible from outside the private network can have an email address in your domain which gets redirected to one of the ISP−supplied email addresses. Others can have local email on the private network. This requires a bit more coordination and configuration, but gives more flexibility than either of the other approaches.

Should you choose to host email for your domain, see section Setting Up Email For Your Domain for configuration help.

If you decide not to host email for your domain, refer to section DNS Configuration If You Are Not Hosting Email for important notes on the name resolution configuration.

6.3 Web Space Hosting

Your ISP may allocate you a certain amount of space on their web servers. You might decide to use that, or you might have a web hosting machine which you put on your external network, in one of your external IP numbers.

Points to remember if you choose to use the ISP's web space hosting:

- You have a certain disk space allocation which you should not exceed. This will include not only web space contents, but also data collected from people visiting the site.
- The bandwidth between your web server and the outside world will almost certainly be higher than it would be if you hosted it on your own hardware. In any case, it will not be slower.
- It may be difficult to install custom CGI scripts or commercial packages on your web site.
- Your bandwidth between your network and your web server will almost certainly be lower than it would be if you hosted it on your own network.

Points to remember if you choose to host your own web space:

- You have much more control over the hosting machine. You can tailor your security more precisely for your application.
- Potentially sensitive data, such as credit card numbers or mailing addresses, remains on machines which you control.
- Your backup strategy is probably not as comprehensive as your ISP's.

Notice that I do not mention anything about the ISP having more powerful hardware, higher peak data rates, and so on. By the time these things become important, you're talking about very high data rate network connections, and, quite frankly, you had better be delegating these decisions to a skilled consultant, not looking in a Linux HOWTO.

Should you choose to host web space for your domain on your own server(s), refer to other documents, such as the WWW−HOWTO, for configuration help. I strongly recommend that this service be run on a different machine from the private network gateway machine, for security reasons.

6.4 FTP Site Hosting

Basically, the same arguments apply to FTP hosting as apply to WWW hosting, with the exception that active content is not an issue for FTP, and CGI scripts don't appear. Most of the recent ftpd exploits have come from buffer overruns resulting from the creation of large directory names in anonymously−writable upload directories, so if your ISP allows uploads and is lax in keeping up with security updates on the FTP daemon, you might be better off hosting this service yourself.

Should you choose to host FTP for your domain on your own server(s), make sure to get the latest version of your FTP daemon, and consult the configuration instructions there. Once more, I strongly recommend that this service be run on a different machine from the private network gateway machine, for security reasons.

For *wu−ftpd*, I would recommend the following configuration options:

- −−disable−upload − unless you need anonymous uploads
- −−enable−anononly − encourage your local users to use *scp* to transfer files between machines.
- −−enable−paranoid − disable whatever features of the current release might be considered questionable.

6.5 Packet Filtering

Some ISPs will put packet filters on their network, to protect the users of the system from each other, or from external attackers. Cable modem networks and similar broadcast networks have had embarrassing problems when users of Windows 95 or 98 inadvertently set up disk shares, exporting the full contents of their hard drives to anybody on the network segment who cared to browse for active servers in the neighbourhood. In some cases, the solution has been to tell the users not to do that, but some providers have put filtering into the access hardware to prevent people from exporting their data by accident.

Packet filtering is really something which you ought to do yourself. It fits in easily into the kernel running on your private network gateway machine and gives you a better idea of what's happening around you. You often will find that you have to make small tweaks to the firewall to optimize it during the initial setup, and this is much easier to do in real time than through a technical support contact.

Should you choose to do packet filtering for your domain, see section Setting Up Packet Filtering for configuration help.

7.Configuring Your Hosted Services

7.1 Setting up Name Resolution

You will want some way for the computers on your network to refer to one another by name, and also a way for people in the outside world to refer to your exposed hosts by name. There are several ways to go about doing this.

DNS On Private Network, ISP Handles Domain

[Note: if you have chosen not to implement a private network, go to section Fully Exposed Network, Hosted By ISP.]

In this configuration, you have delegated responsibility for the primary DNS authority on your domain to the ISP. You still use DNS within your private network when hosts there want to talk to one another. You have given your ISP a list of the names and IP numbers of all exposed hosts. If you want one externally visible machine, for instance betty.example.com, to act both as web and FTP server, you should ask the ISP to make CNAME entries for www.example.com and ftp.example.com pointing to betty.example.com.

Set up DNS on your private network gateway machine. This can be done securely, and makes upgrading easier, should you later decide to host primary DNS authority for your domain.

I will assume that you have decided to host DNS from the machine dns.example.com, which is on the private network gateway, and an alias for fred.example.com at 192.168.2.1. Some small modifications have to be made to this configuration if this is not the case. I will not cover that in this HOWTO unless there is significant interest.

You will have to download and compile a recent version of BIND, the Berkeley Internet Name Domain. It is available at the BIND web site. Next, you have to configure the daemon. Create the following file, `/etc/named.conf`:

```
options {
        directory "/var/named";
        listen-on { 192.168.1.1 };
};

zone "." {
        type hint;
        file "root.hints";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "pz/127.0.0";
};


zone "1.168.192.in-addr.arpa" {
        type master;
        file "pz/1.168.192";
};

zone "example.com" {
        type master;
        notify no;
        file "pz/example.com";
};
```

Note that we are declaring ourselves the master for the example.com domain. Meanwhile, our ISP is also declaring itself to be the master for the same domain. This is not a problem, as long as you are careful about the setup. All of the machines on the private network must use dns.example.com to perform their name resolution. They must *not* use the name resolvers of the ISP, as the ISP name server believes itself to be authoritative over your entire domain, but it doesn't know the IP numbers or names of any machines on your private network. Similarly, hosts on exposed IP numbers in your domain *must* use the ISP name server, not the private name server on dns.example.com.

The various files under `/var/named` must now be created.

The `root.hints` file is exactly as described in the BIND documentation, or in the DNS HOWTO. At the time of this writing, the following is a valid root.hints file:

```
H.ROOT-SERVERS.NET.     6d15h26m24s IN A  128.63.2.53
C.ROOT-SERVERS.NET.     6d15h26m24s IN A  192.33.4.12
G.ROOT-SERVERS.NET.     6d15h26m24s IN A  192.112.36.4
F.ROOT-SERVERS.NET.     6d15h26m24s IN A  192.5.5.241
B.ROOT-SERVERS.NET.     6d15h26m24s IN A  128.9.0.107
J.ROOT-SERVERS.NET.     6d15h26m24s IN A  198.41.0.10
```

```
K.ROOT-SERVERS.NET.     6d15h26m24s IN A  193.0.14.129
L.ROOT-SERVERS.NET.     6d15h26m24s IN A  198.32.64.12
M.ROOT-SERVERS.NET.     6d15h26m24s IN A  202.12.27.33
I.ROOT-SERVERS.NET.     6d15h26m24s IN A  192.36.148.17
E.ROOT-SERVERS.NET.     6d15h26m24s IN A  192.203.230.10
D.ROOT-SERVERS.NET.     6d15h26m24s IN A  128.8.10.90
A.ROOT-SERVERS.NET.     6d15h26m24s IN A  198.41.0.4
```

The `pz/127.0.0` file is as follows:

```
$TTL 86400

@               IN      SOA     example.com. root.example.com. (
                                1       ; Serial
                                8H      ; Refresh
                                2H      ; Retry
                                1W      ; Expire
                                1D)     ; Minimum TTL
                        NS      dns.example.com.
1                       PTR     localhost.
```

The `pz/1.168.192` file is as follows:

```
$TTL 86400

@       IN      SOA             dns.example.com. root.dns.example.com. (
                                1       ; Serial
                                8H      ; Refresh 8 hours
                                2H      ; Retry   2 hours
                                1W      ; Expire  1 week
                                1D      ; Minimum 1 day
                        )
                NS      dns.example.com.

1               PTR     fred.example.com.
                PTR     dns.example.com.
                PTR     mail.example.com.
2               PTR     barney.example.com.
3               PTR     wilma.example.com.
```

and so on, where you create one PTR record for each machine with an interface on the private network. In this example, fred.example.com is on IP number 192.168.1.1, and is pointed to by the dns.example.com and mail.example.com aliases. The machine barney.example.com is on IP number 192.168.1.2, and so on.

The `pz/example.com` file is as follows:

```
$TTL 86400

@               IN      SOA     example.com. root.dns.example.com. (
                                1       ; Serial
                                8H      ; Refresh 8 hours
                                2H      ; Retry   2 hours
                                1W      ; Expire  1 week
                                1D      ; Minimum 1 day
                        )
                        NS              dns.example.com.
        IN              A               192.168.1.1
        IN              MX          10  mail.example.com.
        IN              MX          20  <ISP mail machine IP>.


localhost               A           127.0.0.1
fred                    A           192.168.1.1
                        A           10.1.1.9
dns                     CNAME       fred
mail                    CNAME       fred
barney                  A           192.168.1.2
wilma                   A           192.168.1.3
betty                   A           10.1.1.10
www                     CNAME       betty
ftp                     CNAME       betty
```

Note that we create entries for machines both within the private network and on external IPs, since machines within the private network will not query the ISP's name servers for a request on, say, betty.example.com. We also provide both IP numbers for fred, the private and external IP numbers.

One line in the ``options'' section of `/etc/named.conf` bears discussion:

```
listen-on { 192.168.1.2 };
```

This will prevent your named daemon from answering DNS requests on the outside interface (all requests from the outside must go through the ISP's name resolver, not yours).

Non−DNS Resolution On Private Network, ISP Handles Domain

[Note: if you have chosen not to implement a private network, go to section Fully Exposed Network, Hosted By ISP.]

In this configuration, you have decided that your private network is fairly small and unlikely to change often. You have decided not to use the centralized database of a DNS server, and instead to maintain the host resolution separately on each machine. All machines should use the ISP's DNS server for their host name resolution for machines beyond the private network gateway. For name resolution on the private network, a hosts table has to be created. For Linux, this means entering the names and IP numbers of all of the machines on the private network into the `/etc/hosts` on each machine. Any time a new machine is added, or a name or IP number is changed, this file has to be updated on each Linux box.

As in section DNS Resolution on Private Network, ISP Handles Domain, the list of host names on exposed IP numbers must be sent to the ISP, and any aliases (such as for www and ftp names) should be specified so that a CNAME entry can be created by the ISP.

You Are Primary DNS Authority For Domain

While you could set up *named* resolution on the exposed hosts, and private database resolution for the private network, I will not cover that case. If you're going to be running named for one service, you ought really to do it for both, just to simplify the configuration. In this section I will assume that the private network gateway machine is handling name resolution both for the private network and for outside requests.

At the time of this writing, under version 8.2.2 of the BIND package, there is no way for a single *named* daemon to produce different answers to requests, depending on which interface the request arrives on. We want name resolution to act differently if the query comes from the outside world, because IP numbers on the private network shouldn't be sent out, but have to be available in answer to requests from within the private network. There is some discussion of a new ``views'' keyword which may be added to BIND to fill this need at a later date, but until that happens, the solution is to run two *named* daemons with different configurations.

First, set up the private network domain name server as described in section DNS Resolution on Private Network, ISP Handles Domain. This will be the name resolver visible from within your private network.

Next, you have to set up DNS for your domain, as visible to hosts in the outside world. First, check with your provider to see if they will delegate reverse lookups of your IP numbers to them. While the original DNS standard didn't account for the possibility of controlling reverse DNS on subnets smaller than a class C network, a workaround has been developed which works with all compliant DNS clients, and has been outlined in RFC 2317. If your provider is willing to delegate control of reverse DNS on your IP block, you will have to determine from them the exact name of the in−addr pseudo−domain they have chosen to delegate to (the RFC does not offer a convention they recommend for everyday use), and you will have to register control for that pseudo−domain. I will assume that the provider has delegated control to you, and the name of the pseudo−domain is 8.1.1.10.in−addr.arpa. The provider would create CNAME entries of the form

```
8.1.1.10.in-addr.arpa.     2H IN CNAME 8.8.1.1.10.in-addr.arpa.
9.1.1.10.in-addr.arpa.     2H IN CNAME 9.8.1.1.10.in-addr.arpa.
10.1.1.10.in-addr.arpa.    2H IN CNAME 10.8.1.1.10.in-addr.arpa.
etc.
```

in their zone file for the 1.1.10.in−addr.arpa domain. The configuration of your 8.1.1.10.in−addr.arpa zone file is given later in this section.

If your provider is willing to delegate control of the reverse DNS to you, they will create CNAME entries in their reverse DNS zone table for those IP numbers you control, pointing to the corresponding records in your pseudo−domain, as shown above. If they are not willing to delegate control to you, you will have to ask them to update their reverse DNS entries any time you add, delete, or change the name of an externally visible host in your domain. If the reverse DNS table is not synchronized with your forward DNS entries, certain services may generate warnings, or refuse to handle requests issued by machines affected by the mismatch.

You now have to create a second *named* setup, this one to handle requests issued by machines outside the private network gateway. This setup lists only those hosts and IP numbers which are externally visible, and responds only to requests on the outside interface of the private network gateway machine.

First, create a second configuration file, for instance `/etc/named.ext.conf` for requests from the external interface. In our example, it might be as follows:

```
options {
        directory "/var/named";
        listen−on { 10.1.1.9; };
};

zone "." {
        type hint;
        file "root.hints";
};

zone "0.0.127.in−addr.arpa" {
        type master;
        file "pz/127.0.0";
};


zone "8.1.1.10.in−addr.arpa" {
        type master;
        file "ext/8.1.1.10";
};

zone "example.com" {
        type master;
        notify no;
        file "ext/example.com";
};
```

The `root.hints` and `pz/127.0.0` files, both under `/var/named` are shared with the other running daemon. The file `ext/8.1.1.10` is as follows:

```
$TTL 86400

@       IN      SOA             fred.example.com. root.fred.example.com. (
                                1               ; Serial
                                10800           ; Refresh       3 hours
                                3600            ; Retry         1 hour
```

```
                                3600000         ; Expire        1000 hours
                                86400 )         ; Minimum       24 hours
                NS      dns.example.com.
9       IN      PTR     fred.example.com.
                PTR     dns.example.com.
                PTR     mail.example.com.
10      IN      PTR     betty.example.com.
                PTR     www.example.com.
                PTR     ftp.example.com.
```

The file `ext/example.com` contains the following:

```
$TTL 86400

@               IN      SOA     example.com. root.fred.example.com. (
                                10021   ; Serial
                                8H      ; Refresh 8 hours
                                2H      ; Retry   2 hours
                                1W      ; Expire  1 week
                                1D      ; Minimum 1 day
                        )
                        NS              fred.example.com.
        IN              A               209.217.100.58
        IN              MX          10  mail.example.com.
        IN              MX          20  <ISP Mail Machine>.


localhost               A           127.0.0.1
fred                    A           10.1.1.9
betty                   A           10.1.1.10
dns                     CNAME       fred
mail                    CNAME       fred
www                     CNAME       betty
ftp                     CNAME       betty
```

Start the two daemons on the private network gateway machine. Put the following into your network daemon initialization scripts:

```
/usr/sbin/named −u dnsuser −g dnsgroup /etc/named.conf
/usr/sbin/named −u dnsuser −g dnsgroup /etc/named.ext.conf
```

I've assumed here that you have created the unprivileged user ``dnsuser, and the corresponding unprivileged group ``dnsgroup''. If a bug in bind turns up, which allows an attacker to execute code from within *named*, the attacker will find himself restricted to those operations available to the unprivileged user. The `/var/named` directory and the files within should not be writable by ``dnsuser''.

The machines on the private network must have their name resolution configured to ask dns.example.com (at

IP 192.168.1.1 in our example), while the externally visible machines can either query the network gateway's outside interface (at IP 10.1.1.9 in our example), or the ISP's DNS servers.

Fully Exposed Network, Hosted By ISP

In this configuration, you have chosen to expose all of your hosts. You have a real IP number for each machine in your domain, and you've given your ISP the list of machine names and IP numbers. The ISP has given you at least one IP number for their DNS host(s). Your Linux boxes are now configured for name resolution in /etc/resolv.conf:

```
search example.com
nameserver <DNS host 1>
nameserver <DNS host 2>
```

Windows boxes are configured with the same parameters, in the network settings dialogues.

7.2 DNS Configuration If You Are Not Hosting Email

The configurations described in section Setting Up Name Resolution have MX records pointing to a machine ``mail.example.com''. The MX record with the lowest priority number following tells remote sites where to send email. Other MX records with higher priority numbers are used as backup email receivers. These backups will hold the mail for a certain period of time if the primary email receiver is not able to accept the messages for some reason. In the examples in that section, I have assumed that fred.example.com, under its alias of mail.example.com, is handling email for the domain. If you have chosen to let the ISP handle all of your email hosting, you should change those MX records to point to the appropriate ISP machines. Ask your ISP technical support representative what host names you should use for the MX records in the various files.

7.3 Setting up Electronic Mail

If you have chosen to do full electronic mail hosting for your domain, you'll have to take special actions for email coming from hosts on the private network, and for allowing transparent mail reading from anywhere within the private network. Unless you're careful, messages are likely to sit around for long times if they are waiting on one host, and the intended recipient is logged on another machine. For security reasons, I recommend that the incoming email not be accessible from the externally visible hosts (this might help to discourage a PHB who wants his desktop machine to be on a real IP, then wonders why he gets brought down by a ping of death twice a day). A transparent email sharing system on the private network fairly straight−forward in sendmail. If anybody wants to provide *tested* solutions for other mail handling daemons, I welcome additions.

A Solution Using "sendmail"

In order that email delivered to one host be visible on all machines, the simplest solution is to export the mail spool directory with read−write privileges over the entire private network. The private network gateway machine will also act as mail collector and forwarder for the entire private network, and so must have root write privileges to the mail spool drive. The other clients may or may not squash root, at your discretion. My general security philosophy is not to grant privileges unless there is a clear reason for it, so I squash root on the mail spool network drive for all hosts except the private network gateway machine. This has the effect that root can only read his mail from that machine, but this is not a particularly serious handicap. Note that the mail spool drive can be a directory on the private network gateway machine, exported via NFS, or it can be a directory on one of the internal servers, exported to the entire private network. If the mail spool drive is resident on the private network gateway, there is no issue of squashing root for that machine. If it is on another server, then note that email will be undeliverable if that server, the gateway machine, or the network connecting them, is down.

For Windows machines on your private network, you may either set up a POP server on the mail spool host, or use samba to export the mail spool to those machines. The Windows machines should be configured to send and retrieve mail under a Linux username, such as joeuser@example.com, so that the email address host name is the bare domain name, not a machine name like barney.example.com. The outgoing SMTP host should be set to the private network gateway machine, which will be responsible for forwarding the mail and doing any address rewriting.

Next, you should configure sendmail to forward email from the machines on the private network, rewriting the addresses if necessary. Obtain the latest sources to sendmail from the sendmail.org WWW site. Compile the binaries, then go to the `cf/domain` subdirectory within the sendmail source tree, and create the following new file: `example.com.m4`

```
divert(−1)
#
# Copyright (c) 1998 Sendmail, Inc.  All rights reserved.
# Copyright (c) 1983 Eric P. Allman.  All rights reserved.
# Copyright (c) 1988, 1993
#       The Regents of the University of California.  All rights reserved.
#
# By using this file, you agree to the terms and conditions set
# forth in the LICENSE file which can be found at the top level of
# the sendmail distribution.
#
#

#
#  The following is a generic domain file.  You should be able to
#  use it anywhere.  If you want to customize it, copy it to a file
#  named with your domain and make the edits; then, copy the appropriate
#  .mc files and change `DOMAIN(generic)' to reference your updated domain
#  files.
#
divert(0)
```

```
define(`confFORWARD_PATH', `$z/.forward.$w+$h:$z/.forward+$h:$z/.forward.$w:$z/.forward')dnl
FEATURE(redirect)dnl
MASQUERADE_AS(example.com)dnl
FEATURE(masquerade_envelope)dnl
```

This defines the domain ``example.com''. Next, you have to create the `sendmail.cf` files which will be used on the mail host (the private network gateway), and on the other Linux nodes on the private network.

Create the following file in the sendmail source tree, under `cf/cf`: `example.master.m4`

```
divert(−1)
#
# Copyright (c) 1998 Sendmail, Inc.  All rights reserved.
# Copyright (c) 1983 Eric P. Allman.  All rights reserved.
# Copyright (c) 1988, 1993
#       The Regents of the University of California.  All rights reserved.
#
# By using this file, you agree to the terms and conditions set
# forth in the LICENSE file which can be found at the top level of
# the sendmail distribution.
#
#

#
#  This is the prototype file for a configuration that supports nothing
#  but basic SMTP connections via TCP.
#
#  You MUST change the `OSTYPE' macro to specify the operating system
#  on which this will run; this will set the location of various
#  support files for your operating system environment.  You MAY
#  create a domain file in ../domain and reference it by adding a
#  `DOMAIN' macro after the `OSTYPE' macro.  I recommend that you
#  first copy this to another file name so that new sendmail releases
#  will not trash your changes.
#

divert(0)dnl
OSTYPE(linux)dnl
DOMAIN(example.com)dnl
FEATURE(nouucp)
FEATURE(relay_entire_domain)
FEATURE(`virtusertable', `hash /etc/sendmail/virtusertable')dnl
FEATURE(`genericstable', `hash /etc/sendmail/genericstable')dnl
define(`confPRIVACY_FLAGS', ``noexpn,novrfy'')dnl
MAILER(local)
MAILER(smtp)
Cw fred.example.com
Cw example.com
```

In this example we have disabled the ``expn'' and ``vrfy'' commands. An attacker could troll for aliases with ``expn'', trying names like ``staff'', ``allstaff'', ``office'', and so on, until he hits an alias which expands out several usernames for him. He can then try the usernames against certain weak passwords in hopes of getting in (assuming he can get a login prompt − the security settings described in section Securing Your Domain are set up so that no login prompt is available for off−site attackers).

The other file you should create will define the sendmail.cf for the slave machines: `example.slave.m4`

```
divert(-1)
#
# Copyright (c) 1998 Sendmail, Inc.  All rights reserved.
# Copyright (c) 1983 Eric P. Allman.  All rights reserved.
# Copyright (c) 1988, 1993
#       The Regents of the University of California.  All rights reserved.
#
# By using this file, you agree to the terms and conditions set
# forth in the LICENSE file which can be found at the top level of
# the sendmail distribution.
#
#


#
#  This the prototype for a "null client" -- that is, a client that
#  does nothing except forward all mail to a mail hub.  IT IS NOT
#  USABLE AS IS!!!
#
#  To use this, you MUST use the nullclient feature with the name of
#  the mail hub as its argument.  You MUST also define an `OSTYPE' to
#  define the location of the queue directories and the like.
#  In addition, you MAY select the nocanonify feature.  This causes
#  addresses to be sent unqualified via the SMTP connection; normally
#  they are qualified with the masquerade name, which defaults to the
#  name of the hub machine.
#  Other than these, it should never contain any other lines.
#

divert(0)dnl

OSTYPE(linux)
FEATURE(nullclient, fred.$m)
Cm example.com
```

You build the appropriate sendmail.cf files with the command:

```
make example.master.cf example.slave.cf
```

and then copy the files to the appropriate machines under the name `sendmail.cf`.

This configuration puts most of the sendmail configuration files under the `/etc/sendmail/` subdirectory. This configuration causes *sendmail* to parse and use two special files, `virtusertable.db` and `genericstable.db`. To use these special files, create their parent files. First, `virtusertable.src`:

```
John.Public@example.com                 jpublic
Jane.Doe@example.com                    jdoe@somemachine.somedomain
abuse@example.com                       root
Pointyhaired.Boss@example.com           #phb#@hotmail.com
```

This maps the email addresses on incoming email to new destinations. Mail sent to John.Public@example.com is delivered locally to the Linux account jpublic. Mail to Jane.Doe@example.com is redirected to another email account, possibly in a different domain. Mail to abuse@example.com is sent to root, and so on. The other file is `genericstable.src`:

```
jpublic                                 John.Public@example.com
janedoe                                 Jane.Doe@example.com
whgiii                                  Pointyhaired.Boss@example.com
```

This file renames the sender on outgoing email from locally−sourced mail. While it clearly can't affect the return address for mail sent directly from jdoe@somemachine.somedomain, it allows you to rewrite the sender's email address from the internal usernames to whatever email address convention you've chosen. Finally, create the following `Makefile` in `/etc/sendmail/`:

```
all : genericstable.db virtusertable.db

virtusertable.db : virtusertable.src
        makemap hash virtusertable < virtusertable.src

genericstable.db : genericstable.src
        makemap hash genericstable < genericstable.src
```

Run *make* to create the hashed files which *sendmail* can use, and remember to re−run *make* and restart *sendmail* (or send it a SIGHUP) after any changes to either of these ``.src'' files.

Solutions Using Other Mail Transfer Agents

My experience is only with sendmail. If anybody would like to write this section, please contact me. Otherwise, I may, at some later time, try to provide details myself on such MTAs as *Postfix*, *Exim*, or *smail*. I'd really rather somebody wrote these sections who uses those programs.

7.4 Setting up Web Space Hosting

You should set up your externally visible web server on a machine outside the private network, and not on the private network gateway machine, for security reasons. If the web server needs access to databases or other resources stored on the private network, the situation becomes more complicated, both from a network and a security standpoint. Such configurations are beyond the scope of this document.

The details of setting up the server itself can be found in the apache documentation, and in the Linux WWW HOWTO document.

7.5 Setting up FTP Hosting

Once again, your FTP host should be an externally visible machine, and not the private network gateway machine. Follow the setup directions which ship with your FTP daemon package. Be sure to download the most recent version of the daemon, as there are security vulnerabilities in some older versions of many daemons. If your FTP site does not require anonymous users to upload files, be sure to disable that feature in the daemon. I recommend that user (non−anonymous) FTP logins not be permitted on the FTP host, that you require your regular users to use scp, the secure shell remote copy command, for any file updating they may have to do on the FTP host. This is to help build secure habits in the users, and to protect against the ``hostile router'' problem described in section Securing Your Domain.

7.6 Setting up Packet Filtering

This is discussed in detail in section Configuring Your Firewall.

8.Securing Your Domain

This section deals with setting up security for your new domain. The emphasis is on user−transparent features. If your security is too obtrusive, and interferes strongly with the actions of the users, the users will develop their own workarounds which may compromise the entire domain. The best way to avoid this is to make the security as transparent as possible, and to encourage users to come to you first when they have difficulties which might be related to the security measures of the site. A certain flexibility in attitude is important. I know from personal experience that if the security policy is too rigid, the users will simply set up their own network tunnels through the firewall so they can log in from outside the domain. It's better that remote login procedures, or whatever the users are trying to do, be set up, inspected, and approved by you.

This section deals with securing your network against outside attack, and against casual snooping from within. Securing your site against determined attack from validated users within the private network is a more difficult and involved task, and is beyond the scope of this document.

One of the security considerations used in this section is protecting against the ``hostile router''. The router provided by your ISP may be a remotely configurable computer in its own right, with the administrative password held by your provider. There have been security problems in the past when the router's manufacturer override password (the one used when your ISP forgets the password they programmed in) has become known to system crackers. When possible, you should design your security around the assumption that the router is potentially hostile. That is, it could be using any IP number in your public *or private* network blocks, it could be redirecting traffic on outgoing packets to another site, and it could be recording anything which goes through.

8.1 Configuring Your Firewall

This section deals with configuring an *ipchains*−based masquerading, forwarding, filtering router. You should probably read the IPCHAINS−HOWTO document first, then look here for additional hints. That HOWTO describes the steps necessary to compile a kernel with masquerading support, and describes the use of the ipchains binary in detail. You should enable firewalling on all machines with exposed IP numbers.

Check your startup scripts to make sure that the sequence is as follows on the private network gateway machine:

1. Outside Ethernet card is initialized.
2. Firewall rules are run through ipchains.
3. Forwarding is turned on.
4. Network service daemons are started.

So, as an example, on a Slackware−based system, the firewall configuration should come between the execution of `rc.inet1` and `rc.inet2`. Further, if any problems arise during the firewall configuration steps, a warning should be printed, and the external Ethernet card taken off line before the network service daemons are run.

One common problem with ipchains−based firewalls is the tedium of making sure that your rules are correctly set for packets arriving from the loopback interface, or arriving from either of the internal or external interfaces on the firewall machine. These locally−sourced packets can be blocked by a firewall. All too often, this is fixed by a sort of shotgun debugging approach, whereby the rules for the firewall are tweaked until all applications seem to run properly on the firewall host again. Unfortunately, this can sometimes result in a firewall which has unintended holes. With ipchains it is possible to write a firewall script which is easily debugged, and which avoids many of the packet source problems. Here is a sample script, `/sbin/firewall.sh`:

```
#! /bin/sh
#
# New firewalling script using IP chains. Creates a filtering router
# with network masquerading.
#

# define a few variables

IPCHAINS=/sbin/ipchains

LOCALNET="192.168.1.0/24"       # the private network
ETHINSIDE="192.168.1.1"         # fred.example.com's private IP #
```

```
ETHOUTSIDE="10.1.1.9"           # fred.example.com's public IP #
LOOPBACK="127.0.0.1/8"
ANYWHERE="0/0"
OUTSIDEIF=eth1                  # fred.example.com's private interface

FORWARD_PROCENTRY=/proc/sys/net/ipv4/ip_forward

#
# These two commands will return error codes if the rules
# already exist (which happens if you run the firewall
# script more than once). We put the commands before "set -e"
# so that the script doesn't abort in that case.

$IPCHAINS -N outside
$IPCHAINS -N portmap

set -e                  # Abort immediately on error setting
                        # up the rules.


#
# Turn off forwarding and clear the tables

echo "0" > ${FORWARD_PROCENTRY}

$IPCHAINS -F forward
$IPCHAINS -F input
$IPCHAINS -F output
$IPCHAINS -F outside
$IPCHAINS -F portmap


#
# Masquerade packets from within our local network destined for the
# outside world. Don't masquerade packets which are local to local

$IPCHAINS -A forward -s $LOCALNET -d $LOCALNET -j ACCEPT
$IPCHAINS -A forward -s $ETHOUTSIDE -d $ANYWHERE -j ACCEPT
$IPCHAINS -A forward -s $LOCALNET -d $ANYWHERE -j MASQ

#
# Set the priority flags. Minimum delay connections for www, telnet,
# ftp, and ssh (outgoing packets only).

$IPCHAINS -A output -p tcp -d $ANYWHERE www -t 0x01 0x10
$IPCHAINS -A output -p tcp -d $ANYWHERE telnet -t 0x01 0x10
$IPCHAINS -A output -p tcp -d $ANYWHERE ftp -t 0x01 0x10
$IPCHAINS -A output -p tcp -d $ANYWHERE ssh -t 0x01 0x10


#
# Anything from our local class C is to be accepted, as are
# packets from the loopback and fred's external IP.

$IPCHAINS -A input -s $LOCALNET -j ACCEPT
$IPCHAINS -A input -s $LOOPBACK -j ACCEPT
$IPCHAINS -A input -s $ETHOUTSIDE -j ACCEPT



# We'll create a set of rules for packets coming from the big, bad
# outside world, and then bind all external interfaces to it. This
```

```
# rule will be called "outside"
#
# We also create a "portmap" chain. The sockets used by daemons
# registered with the RPC portmapper are not fixed, and so it is
# a bit difficult to set up filter rules for them. The portmap
# chain is configured in a separate script.


#
# Send packets from any outside interface to the "outside"
# rules chain. This includes the $OUTSIDEIF interface and any
# ppp interfaces we create for dialout (or dialin).

$IPCHAINS −A input −i ${OUTSIDEIF} −j outside
$IPCHAINS −A input −i ppp+ −j outside


##################################################
#
#  Set up the "outside" rules chain              #
#
##################################################

#
# Nobody from the outside should claim to be coming from our localnet
# or loopback

$IPCHAINS −A outside −s $LOCALNET −j DENY
$IPCHAINS −A outside −s $LOOPBACK −j DENY

#
# No packets routed to our local net should come in from outside
# because the outside isn't supposed to know about our private
#  IP numbers.

$IPCHAINS −A outside −d $LOCALNET −j DENY

#
# Block incoming connections on the X port. Block 6000 to 6010.

$IPCHAINS −l −A outside −p TCP −s $ANYWHERE −d $ANYWHERE 6000:6010 −j DENY

#
# Block NFS ports 111 and 2049

$IPCHAINS −l −A outside −p TCP −s $ANYWHERE −d $ANYWHERE 111 −j DENY
$IPCHAINS −l −A outside −p TCP −s $ANYWHERE −d $ANYWHERE 2049 −j DENY
$IPCHAINS −l −A outside −p UDP −s $ANYWHERE −d $ANYWHERE 111 −j DENY
$IPCHAINS −l −A outside −p UDP −s $ANYWHERE −d $ANYWHERE 2049 −j DENY

#
# Block XDM packets from outside, port 177 UDP

$IPCHAINS −l −A outside −p UDP −s $ANYWHERE −d $ANYWHERE 177 −j DENY


#
# Block the YP/NIS port 653

$IPCHAINS −l −A outside −p TCP −s $ANYWHERE −d $ANYWHERE 653 −j DENY

#
```

```
# Don't bother logging accesses on TCP port 80, the www port.

$IPCHAINS -A outside -p TCP -s $ANYWHERE -d $ANYWHERE 80 -j DENY

#
# Accept FTP data and control connections.

$IPCHAINS -A outside -p TCP -s $ANYWHERE 20:21 -d $ANYWHERE 1024: -j ACCEPT

#
# Accept ssh packets

$IPCHAINS -A outside -p TCP -s $ANYWHERE -d $ANYWHERE ssh -j ACCEPT

#
# Accept DNS packets from outside

$IPCHAINS -A outside -p TCP -s $ANYWHERE -d $ANYWHERE 53 -j ACCEPT
$IPCHAINS -A outside -p UDP -s $ANYWHERE -d $ANYWHERE 53 -j ACCEPT

#
# Accept SMTP from the world

$IPCHAINS -A outside -p TCP -s $ANYWHERE -d $ANYWHERE 25 -j ACCEPT

#
# Accept NTP packets

$IPCHAINS -A outside -p UDP -s $ANYWHERE -d $ANYWHERE 123 -j ACCEPT

#
# Accept no tap ident packets, we don't use them

$IPCHAINS -A outside -p TCP -s $ANYWHERE -d $ANYWHERE 113 -j DENY

#
# Turn off and log all other packets incoming, TCP or UDP, on privileged ports

$IPCHAINS -l -A outside -p TCP -s $ANYWHERE -d $ANYWHERE :1023 -y -j DENY
$IPCHAINS -l -A outside -p UDP -s $ANYWHERE -d $ANYWHERE :1023 -j DENY

#
# Check against the portmapper ruleset

$IPCHAINS -A outside -j portmap


##############################################
#
#    End of "outside" rules chain            #
#
##############################################


#
# Block outgoing rwho packets

$IPCHAINS -A output -p UDP -i $OUTSIDEIF -s $ANYWHERE 513 -d $ANYWHERE -j DENY

#
# Prevent netbios packets from leaving
```

```
$IPCHAINS -A output -p UDP -i $OUTSIDEIF -s $ANYWHERE 137 -d $ANYWHERE -j DENY

#
# Turn on forwarding

echo "1" > ${FORWARD_PROCENTRY}
```

Notice that the firewall can be used not only to block incoming packets, but also outgoing packets which might leak information about your private network, such as rwho and netbios packets.

As noted earlier, the portmapper rules are a bit different, because the portmap daemons register themselves with the portmapper and are told which ports to listen on. The ports used by a particular daemon may change as you change the RPC services used, or change their order of startup. The following script, `/sbin/firewall.portmap.sh` generates rule sets for the portmapped daemons:

```
#! /bin/sh
#
ANYWHERE=0/0

IPCHAINS=/sbin/ipchains

$IPCHAINS -F portmap

# Rules for preventing access to portmapped services by people on the outside
#
/usr/bin/rpcinfo -p | tail +2 | \
        { while read program vers proto port remainder
          do
                prot=`echo $proto | tr "a-z" "A-Z"`
                $IPCHAINS -l -A portmap -p $prot -s $ANYWHERE -d $ANYWHERE $port -j DENY || ex
          done
        }
```

We didn't have to worry about whether packets coming in were legitimate packets from the private network, the portmap chain is only checked when the packets come in from the outside.

This firewall configuration logs most suspicious packets through klogd with the kern.info logging priority. It will log normal connection attempts, as well as all known ``stealth'' probes.

Now, we put these all together. We'd like to make sure that there isn't a small window of vulnerability while the system is starting up, so you should configure your startup sequence as follows:

```
#! /bin/sh
#
# Get the network started, securely
#
#
/etc/rc.d/rc.inet1              # Configure the network interfaces
                                # and set up routing.
```

```
/sbin/firewall.sh || { echo "Firewall configuration failed"
                       /sbin/ifconfig eth1 down }

/sbin/ipchains −I outside 1 −j DENY     # Deny all incoming packets

/etc/rc.d/rc.inet2              # Start the network daemons

sleep 5                         # Let them stabilize

# Secure the portmapped services
/sbin/firewall.portmap.sh || { echo "Portmap firewall configuration failed"
                               /sbin/ifconfig eth1 down }

/sbin/ipchains −D outside 1       # Allow incoming packets
```

This assumes that eth1 is the interface on the externally visible IP number. If any of the ipchains rule sets fail to install, a warning is issued and that interface is taken off line. The ``outside'' chain is set to deny all packets before the network service daemons are started, because the firewalling rules are not yet in place for the portmapped services. Once the portmapped services are firewalled, the ``outside'' chain is restored to its proper behaviour.

8.2 Configuring SSH1

At the time of this writing, OpenSSH (v1.2pre17) doesn't offer one of the features I mention here, but OpenSSH is under very active development, and this could change at any time. The particular feature which is missing is a configuration setting which allows you to insert *scp*, *ssh*, and *slogin* as binaries named *rcp*, *rsh*, and *rlogin*, with transparent fall−through in the ssh client programs to the original *rsh*, *rcp*, or *rlogin* when the remote site isn't running *sshd*. Making an invocation of *rsh* run, instead, the *ssh* client program is, in my opinion, important for keeping the security easy to use and out of the way of the users. Everybody's scripts, *rdist* configurations, and so on will continue to work without modification if the remote site is running sshd, but data will be sent encrypted, with strong host authentication.

Obtain ssh1 from the ssh web site, and compile it to replace the unencrypted r−programs (*rsh*, *rlogin*, and *rcp*). First, copy those three files to `/usr/lib/rsh/`, then configure the ssh package with:

```
./configure −−with−rsh=/usr/lib/rsh/rsh −−program−transform−name='s/^s/r/' −−prefix=/usr
```

Install the binaries, and configure according to the directions. On the private network gateway machine, make sure that the sshd configuration has the following entries defined:

```
ListenAddress 192.168.1.1       # fred's internal IP
IgnoreRhosts no
X11Forwarding yes
X11DisplayOffset 10
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
```

You will have to do further configuration of other entries in the `/etc/sshd_config` file, but try not to change these fields. Once you have all of the entries in the file set to your satisfaction, copy this entire file into a new file, `/etc/sshd_config.ext`, for the external network. Change two fields in the new file: the ``ListenAddress'' should be changed to the private network gateway's external IP number (10.1.1.9 in our fred.example.com case), and ``PasswordAuthentication'' should be set to ``no'' in `/etc/sshd_config.ext`. In your network services startup script, start sshd twice, once with

```
/usr/sbin/sshd
```

and once with

```
/usr/sbin/sshd −f /etc/sshd_config.ext
```

This will create two running sshd daemons. The one operating on the internal interface will allow logins with passwords, but the external interface will require an RSA key validation before anybody can log on.

Next, turn off incoming telnet and shell services in the inetd configuration file (note that the firewall configuration listed in section Configuring Your Firewall already prevents access from outside, but it's best to defend in depth, don't rely on everything working correctly).

People who want to be able to log in from home, or from out of town, will need an RSA key. Make sure they know how to do this, so they don't spend their energies trying to figure out another way to do it, like running a telnetd on an unprivileged port on your firewall machine.

An RSA key is generated by the command:

```
ssh−keygen −b 1024 −f new_rsa_key
```

You will be prompted for a pass phrase. This should *not* be blank. A person with access to the file `new_rsa_key`, and knowledge of the pass phrase, has everything necessary to pass an RSA authentication challenge. The pass phrase can be an ``unguessable'' password, or a long sentence, but make it something non−trivial. The file `new_rsa_key` can be copied to a floppy disk, or onto a laptop, and, along with the pass phrase, can be used to log into accounts which are set to grant access to that particular RSA key.

To configure an account to allow access by a particular RSA key, simply create a `$HOME/.ssh/` directory for that user on the private network gateway machine (i.e. the machine which will be receiving the login attempt), and copy the file `new_rsa_key.pub` which was created by the "ssh−keygen" command into the file `$HOME/.ssh/authorized_keys`. See the section ``AUTHORIZED_KEYS FILE FORMAT'' in the sshd man page for details on other options you can add to the key, such as requiring the login to come from a certain IP or host name, or authorizing the key only to permit the remote invocation of certain commands (for instance, an RSA key which commands a backup to take place, or commands a status report to be emailed somewhere off site).

Only one thing remains to make the RSA key mechanism as gentle as possible to the users. If a user is forced to enter the pass phrase more than once or twice in a session, they are likely to become bored and take security matters into their own hands. Under Linux, arrange their login shell to be invoked under *ssh−agent*. For instance, if the company laptop used on business trips runs *xdm*, and drops users into an X session, go into the `/var/X11R6/lib/xdm/Xsession_0` file and change the lines which invoke the startup, which are probably of the form:

```
exec "$startup"
```

into lines of the form:

```
exec ssh-agent "$startup"
```

In my xdm setup, there are three such lines which should be altered in that one file. Now, when the user logs onto the laptop, he enters the command

```
ssh-add new_rsa_key
```

at any prompt, enters the pass phrase when prompted, and all windows will have pass phrase−free access to the account on the private network gateway until the user logs off his X session on the laptop.

Run sshd on all of the machines on your private network, as well as on any exposed hosts. For machines other than the private network gateway machine, the ListenAddress entry in `/etc/sshd_config` can be set to ``0.0.0.0''. You should set up the host keys with the command:

```
ssh-keygen -b 1024 -f /etc/ssh_host_key -N ""
```

then run *make−ssh−known−hosts* and distribute the `/etc/ssh_known_hosts` file among all of the machines on the private and public networks.

Disable incoming telnet and the unencrypted r−services. Don't delete the *telnet* binary, it's useful for things other than simple telnet sessions on port 23. You should allow password authentication on the private network, and disable it on the exposed machines, requiring an RSA key to log onto the exposed hosts.

It is convenient for the users if the hosts on the private network are mentioned in each other's `/etc/hosts.equiv` files. The sshd daemons will respect those, and allow people to rlogin and rsh between machines without passwords or pass phrases. On every connection, the machines will be verifying each other's identities with host−level RSA keys.

One difficulty arises when a user logged onto a machine on the private network wants to log onto a box on an exposed IP number. You can't use `/etc/hosts.equiv` or `$HOME/.shosts` to allow password−less validation, because the user is coming from a machine whose IP number cannot be determined − it will appear to be coming from the masquerading firewall machine, but the host keys won't match. There are two solutions to this. First, if you insist on using the `/etc/hosts.equiv` or `$HOME/.shosts` methods, the user will have to log onto the private network gateway machine (fred.example.com in our example here), and then log through to the exposed machine from there. The other technique is to use RSA key authentication, that always works regardless of what games are going on with IP numbers and host name lookups.

8.3 Configuring X

In the user's continuing quest to prove that he values convenience over security, it has become common for people to put

```
xhost +
```

commands right into their X initialization scripts. This grants X server access to everybody in the world. Now the random outsider can change your root window graphic to something embarrassing while your boss is showing his mother around your office. Alternately, this outsider can quietly monitor every keystroke you issue, and dump the contents of your screen to his desktop. Needless to say, this doesn't bode well for passwords used to log into other sites, or for sensitive documents being edited on screen. The xhost protocol itself is inherently limited, as it is not possible to grant permissions to use the screen on a user basis, only on a machine basis.

Enter *xauth* authentication. If you have *xdm* you probably already are running *xauth* authentication, but *xhost* still works, and might still be what people are using to run X processes between machines. Once again, the goal is to make the security easy enough to use that the users aren't tempted to run the *xhost* command anymore.

The sshd setup described in section Configuring SSH1, with the ``X11Forwarding'' flag set, is actually simpler to use than the *xhost* technique. Once you have logged into your terminal, you can simply rlogin to a remote machine, and run *netscape*, *xv*, or whatever you like, without having to set the $DISPLAY variable name or allow explicit permissions. During *ssh* login, it configures the system in a way transparent to the end user, and even encrypts all of your X packets before they go over the network.

If you are unable to use the sshd X11 forwarding for some reason, you should use *xauth* when you want to authorize other machines to have access to your X server. Document this for the users, or create specialized shell scripts to help them out. The relevant command to authorize a particular login, ``jpublic'', on machine ``barney'' to have access to your X server is:

```
/usr/X11/bin/xauth extract − $DISPLAY | rsh −l jpublic barney /usr/X11/bin/xauth merge −
```

I'd be tempted to delete *xhost* from your machines entirely. If it causes problems with any programs, you will at least know that those programs had poorly−designed security. It's simple enough to build a shell script as a drop−in replacement for *xhost* which uses the *xauth* sequence listed above.

Note that if *rsh* is not the encrypting ssh program, the xauth key is sent plaintext. Anybody who holds the plaintext of the key can access your server, so you do not gain much security if you don't use ssh for these transactions. Note, also, that if the users' home directories are exported via NFS (the Network File System), the xauth key is available in plaintext to anybody able to snoop those NFS packets, regardless of whether you're running ssh on your systems.

8.4 Configuring Disk Sharing

With email coming to a central machine, the read/send from any host setup described here is very convenient, but some care has to be taken to protect against trivial snooping by bored local users. NFS without AUTH_DES implemented is inherently insecure. NFS relies on the client machine to authenticate access, there is no password verification on the server to make sure that the client should be permitted to access the private files of a particular user. A Windows box can be configured to read NFS−exported volumes as any numeric uid, completely bypassing UNIX file permissions. Consequently, NFS exports should only be made to machines which are always Linux (or UNIX) boxes under your direct control, and never ones which can be dual−booted into Windows. If you want to export the mail spool directory, or any other directory, to machines which can sometimes be used as Windows boxes, export them with samba, setting the authentication mode to ``security=USER''. Connecting the machines on your network with a switch rather than a hub will also help, as it leaves very little of interest for sniffers on Windows machines. Ultimately, though, it's very difficult to secure any disk sharing over the network at the time of this writing.

Why bother, if you can't really secure the network disks? Mostly it's an issue of credible defense. If you leave a sheet of paper on your desk with confidential information, and somebody in the office reads it, he can argue that he didn't realize what the paper was, his natural curiosity just got the better of him when he saw it sitting on the desk. If the sheet of paper were in a filing cabinet or desk drawer, it's an entirely different story. The purpose of taking some basic network security measures internally is to ensure that nobody ``accidentally'' compromises security.

9.Acknowledgements

This document was written as internal documentation for the DYNACAN project, as part of the project's continuing development under the control of the Ministry of Human Resources Development Canada.

This document has benefited considerably from the suggestions of

- Rod Smith (rodsmith@rodsbooks.com), who suggested I provide details on registering a domain name and on setting up with a dynamic IP, and pointed me at the various dynamic IP hosting services and at Granite Canyon.
- Greg Leblanc (gleblanc@my−deja.com) for useful suggestions on improving the clarity of the document.
- Sami Yousif (syousif@iname.com), who told me about www.dhs.org.

10.Glossary of Terms

This is a list of the meanings of some of the words and acronyms used in this document.

CGI Script

A Common Gateway Interface Script. This is a program which is run on demand to generate the content of a web page. If a web page has to do more than simply feed an unchanging text and graphics display to the viewer, you will probably need some sort of dynamic content generation program such as a CGI Script. Examples include discussion boards, feedback forms, e−commerce shopping carts, and more.

DHCP

Dynamic Host Configuration Protocol. A standard, defined in RFC 1531, for computers on a TCP/IP network to request from a central server information such as the IP number they should be using, the netmask, the gateway, etc. Rather than an administrator entering this information into the machine configuration, the machine simply requests it from the server as it is preparing to attach to the network.

DNS

Domain Name Service. A standard for translating domain names into IP Numbers, or vice versa, by looking up data in centralized databases.

DSL

Digital Subscriber Line. A relatively high speed network connection, usually delivered through specialized telephone wiring.

Dynamic IP Number

An IP Number which is assigned periodically or on a per−session basis. No guarantee is made that the number will remain constant. A dynamic IP number might change only when your network connection hangs up and reconnects, or it might change periodically under DHCP negotiation. Certain session−based services such as *telnet* and *ssh* will stop working if the IP number of either end of the connection is changed during the session.

Forward DNS Query

A DNS query which converts a domain name into an IP Number.

FTP

The File Transfer Protocol. A standard system for sending files between machines over the Internet.

ftpd

The daemon responsible for providing FTP services on a host. It responds to queries initiated by a remote client.

Internet Service Provider

See ISP.

IP

See IP Number.

IP Number

The ``address'' of a certain network interface. Under the current addressing standard, called ipv4, this number consists of four 8−bit values, generally written as base−10 numbers separated by dots. Communication between computers on the Internet is based on packets of information sent between IP numbers.

ISP

Internet Service Provider. The company which provides your network connectivity, including connection hardware, service hosting, and leasing out the IP numbers under their control.

Masquerading

A form of filtering in which packets from one machine to the outside world have their headers rewritten so that they appear to come from an intermediate machine. That intermediate machine then passes responses back to the originating machine. The net effect is that an entire network of machines can appear to use a single IP number, that of the masquerading host, for the purpose of outgoing connections.

named

The name server daemon. This is the daemon which answers DNS queries, and is distributed as part of the BIND package.

Network Time Protocol

See NTP.

NTP

Network Time Protocol. A standard for synchronizing your system clock with the ``true time'', defined as the average of many high−accuracy clocks around the world.

OS

Operating system. Linux, Windows, FreeBSD, BeOS, HP−UX, etc.

PHB

Pointy−Haired Boss. A creation of Scott Adams, of Dilbert fame.

Provider

See ISP.

Reverse DNS Query

A DNS query which converts a IP Number into a domain name.

Router

A specialized hardware device which implements rules for where to send packets based on their IP Numbers, and which bridges between your Ethernet hardware and whatever communications medium connects you to your ISP.

ssh

The secure shell. A cryptographically strong replacement for *rlogin*, *telnet*, *ftp*, and other programs. Protects against ``spoofing'', man in the middle attacks, and packet sniffing.

Static IP Number

An IP Number which has been assigned or leased to you permanently. Barring revocation of the agreement which granted you this number, that IP number will always be available for your use, and no other machine on the Internet is allowed to use that number. Contrast this with Dynamic IP Numbers.
