

Put Your In-house Server on the Net

You can develop intranet applications for a corporate organisation using a low-cost backbone. But you must first learn how to manage content, set up e-mail and instant messaging, and secure all these.



This article is about building the following popular intranet applications on this server, which will provide 24 × 7 connectivity. It deals with:

- Setting up a local Web/intranet server.
- Virtual hosting
- Securing Web content and directories
- E-mail server
- Bringing up a corporate instant messaging environment
- Securing this server via a firewall implementation

Let us first try to understand how we are going to place the server in the network so that it is accessible from outside as well as within the network in a secure manner.

A server class machine is recommended for the intranet server. We will have two LAN cards on the server. On *Eth0* (first LAN card) let us map our global IP and on *Eth1* we map our local IP. For security reasons, you should keep your global and local segments separate. You can do this by placing an Ethernet hub between the router and the server. Terminate the router's LAN port and *Eth0* on the hub. *Eth1* can terminate on the internal switch or the hub.

A better way of doing this is to set up three zones—the global zone, DMZ (demilitarised zone) and the local zone. We can even use a low-end machine for a firewall. The e-mail

server, Web server, intranet server or any other server that needs to be accessed from outside as well as from inside the network, should be placed in the DMZ. The firewall server should have three LAN cards. The LAN port of the router should get

terminated on a hub placed between the router and firewall server. One of the LAN cards should have global IP, the second LAN can be the DMZ segment and the third LAN card to server can be the local segment and have a local IP. This is the preferred implementation, but in our case let us have all firewall and intranet applications on the same server.

IMPLEMENTATION OF APACHE WEB SERVER

In most of the Redhat based installations, the Apache Web server is already pre-configured with perl, php, mysql, or cgi. The service, which is responsible for Apache Web server, is *httpd* and can be controlled by */etc/rc.d/init.d/httpd (start|stop|reload|restart|status)*. The most important are the files governing the server. These are:

```
/etc/httpd/conf/httpd.conf—Long file; almost has everything controlling the Web server parameters.
/var/log/httpd—Directory containing the log files, such as error.log and access.log.
/var/www/html—Document root of the Apache Web server (in case of Redhat 7.x and above) /home/httpd/html in case of Redhat 6.x and below.
/var/www/cgi-bin — Cgi directory for the document root.
```

Please note that most of the general modifications to the server's specifications are done in the *httpd.conf* file. Start your Apache Web server with */etc/rc.d/init.d/httpd start*. Point your client machine's Web browser to the IP address of the Web server and you should be having what is shown in Figure 1.

Once your Apache Web server is accessible, you are ready to host your intranet with static or dynamic pages. Let's see how simple text can be uploaded to the server's document root to be displayed as HTML page. In any Text editor,



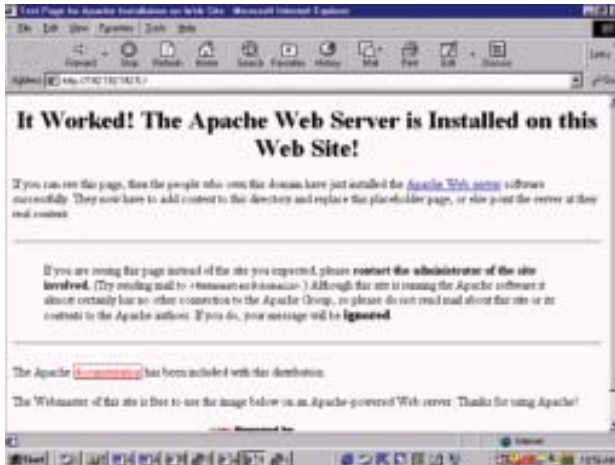


Figure 1

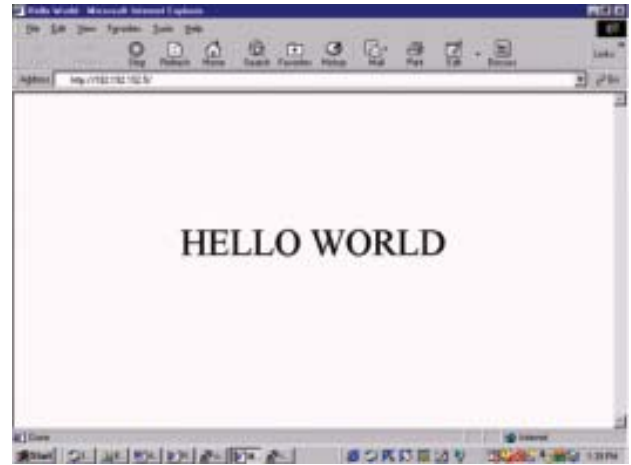


Figure 2

```
<html>
<head>
<title>Hello World </title>
</head>
<body>
<p align="center">&nbsp;&nbsp;&nbsp;</p>
<p align="center">&nbsp;&nbsp;&nbsp;</p>
<p align="center">&nbsp;&nbsp;&nbsp;</p>
<p align="center">&nbsp;&nbsp;&nbsp;</p>
<p align="center"><big><big><big><big><big><big>HELLO
WORLD</big></big></big></big></big></big></p>
</body>
</html>
```

Or you can make HTML pages using tools such as Textpad or Frontpage. Upload this file to the document root of the Web server, i.e., in `/var/www/html/` as `index.html`, replacing the original file. Figure 2 will be displayed on your client's Web browser.

If you are comfortable with PHP, then you could make a script like this:

```
<html>
<head>
<title>PHP</title>
</head>
<body>
<?php echo "Hello World<p>"; ?>
</body>
</html>
```

Upload this file as `index.php` on to the document root and you have the Web page on the browser.

Using the perl scripts we can have the same output

```
#!/usr/bin/perl
print"content-type:text/html\n\n";
print<<'print_html_data';
<html>
<title>Apache</title>
print_html_data
print"Hello World";
print<<"print_html_data";
</html>
print_html_data
```

Save this text in a file that has a `.pl` extension, such as `test.pl`, and upload in the `/var/www/cgi-bin/` directory. Point your client's browser to `http://server_ip_address/cgi-bin/test.pl`.

The environments discussed here were more like static pages. Let's see how we can have dynamic environments using databases. Database engines such as MySQL and postgresql are already in use with Red Hat installations (Redhat 7.x and above), and can be used to build up dynamic intranets.

How do you get 'Hello World' from a database such as MySQL? Before proceeding, you will have to set up the MySQL server, which, in case of Redhat 7.x, can start with:

```
/etc/rc.d/init.d/mysql start
```

If this is the first time you have activated the MySQL server, there will be no password for the MySQL user. You can specify the password via `mysqladmin` command. Activate the MySQL prompt by

```
#mysql -uroot -p (in case you have a password) or mysql
# mysql -p
Enter password:
Welcome to the MySQL monitor. Commands end with; or \g.
Your MySQL connection id is 19 to server version: 4.0.0-alpha
```

Type 'help' for help.

```
mysql>create database test;
Query OK, 1 row affected (0.00 sec)
```

The database test has been created. Now, you'll have to insert a table and a record 'Hello World'.

To connect to test database,

```
mysql> connect test
Connection id: 22
Current database: test
```

To create a table called 'hello'

```
mysql> create table hello(description char(200));
Query OK, 0 rows affected (0.04 sec)
```

To insert a record 'Hello World' in table 'hello'

```
mysql> insert into hello values('hello world');
Query OK, 1 row affected (0.00 sec)

mysql> select * from hello ;
+-----+
| description |
+-----+
| hello world |
+-----+
1 rows in set (0.00 sec)

mysql>
```

You can create a perl script as follows:

```
#!/usr/bin/perl
use DBI;
print"content-type:text/html\n\n";
$dc=DBI->connect("DBI:mysql:test:localhost",root,intranet);
$statement="select * from hello ";
$dcpl = $dc->prepare($statement);
$dcel=$dcpl->execute;
while($hrl=$dcpl->fetchrow_hashref)
{
print" $hrl->{'description'}";
}
```

Please note that test is the database name, localhost is the hostname, root is the username and intranet is the password. Save this file as test.pl with read and execute access in the cgi-bin directory of the document root. Point your client's browser to http://server_ip/cgi-bin/test.pl and you will get 'Hello World' on the browser. This way, you can integrate the database engine to build a dynamic site.

VIRTUAL HOSTING OF INTRANETS ON APACHE

We can have multiple IP-based sites on the same Apache server. This is managed and done by configuring the Apache Web server to support the virtual site hostings. We can have two approaches to configure Apache to support virtual Web hosting. The significance of such a configuration is to have:

1. Multiple intranets on the same server
2. Differentiation among the secure and public areas
3. Multi-purpose hosting and applications running on the same server
4. Defining the local and global Intranets
5. Using the same database applications or the server

The first situation is that you have only one IP available on the Apache you have just tested. This automatically means that whenever you approach this Web server via the Web browser, you will be thrown to the default page. So the question is how do you make the server deliver a page, which is for a different site and still on the same server. That is what virtual hosting is about.

Before practically implementing it, you need to know the

concept that makes it work. So, let us make multiple websites or intranets work on the same IP. For this, we have to take an approach where DNS also plays an important role. In case of reaching the Web server, the client browser does not generally use an IP address. Instead, it uses an address or name, for example, http://mail.linuxforyou.com and not http://203.22.204.104. So DNS comes into the picture—since name resolution is its responsibility. Let's take an instance where we want to host www.linuxforyou.com and www.linuxforyou.org on the same IP or on the same server (as we have only one IP for the server). First, both addresses must point to the same IP on the DNS record. There will be a scenario where you have implemented the DNS or the DNS is somewhere else. In case you have control over DNS, this can be done without much pain. Otherwise, chase the service provider to do the same.

To give you a basic idea, if we have implemented our own DNS, the procedure to have your Linux or named server records will be as follows.

The /etc/named.conf file will contain the following lines:

```
options {
    directory "/var/named";
};
zone "." {
    type hint;
    file "named.ca";
};
zone "linuxforyou.com"{
    type master;
    file "named.linuxforyou.com.forward";
};
zone "linuxforyou.org"{
    type master;
    file "named.linuxforyou.org.forward";
};
```

The file /var/named/named.linuxforyou.com.forward will look like

```
$ttl 38400
linuxforyou.com.      IN      SOA      Intranet.linuxforyou.com.
admin_blore.linuxforyou.com. (
1022485545
10800
3600
604800
38400 )
linuxforyou.com.     IN      NS
Intranet.linuxforyou.com.
mail.linuxforyou.com. IN      A      203.122.252.26
www.linuxforyou.com. IN      A      203.122.252.26
linuxforyou.com.     IN      A      203.122.252.26
```

and the file /var/named/named.linuxforyou.org.forward will look like

```
$ttl 38400
linuxforyou.com.      IN      SOA      Intranet.linuxforyou.com.
admin_blore.linuxforyou.com. (
1022485545
10800
3600
604800
38400 )
linuxforyou.org.     IN      NS
Intranet.linuxforyou.com.
mail.linuxforyou.org. IN      A      203.122.252.26
www.linuxforyou.org. IN      A      203.122.252.26
linuxforyou.org.     IN      A      203.122.252.26
```

Please note that intranet .linuxforyou.com is the FQDN (fully qualified domain name) and is also the acting primary DNS. Now you will see that both the domains point to the intranet server. You are ready to make both the sites work. To do that, you have to specify and decide certain parameters such as document root, location for Error logs and ScriptAlias (for cgi-bin directory). Document root should be the directory or the path where your website's files are physically placed. In this case, it is /home/Intranet/www.linuxforyou.com for the www.linuxforyou.com and /home/Intranet/www.linuxforyou.org for www.linuxforyou.org. For making the perl or scripts run, you need to define your ScriptAlias directory. Otherwise, any script will get redirected to the document root's cgi-bin directory, instead of the virtual host's cgi-bin directory. So, if you are using a perl-based script or database connectivity, make sure you have decided and implemented the same in case of virtual hosts.

Now add the following lines in your httpd.conf file:

```
<VirtualHost www.linuxforyou.com>
DocumentRoot /home/Intranet/www.linuxforyou.com
ServerName www.linuxforyou.com
ScriptAlias /cgi-bin/ /home/Intranet/www.linuxforyou.com/cgi-
bin/
ErrorLog /home/Intranet/www.linuxforyou.com/errlog
</VirtualHost>
<VirtualHost www.linuxforyou.org>
DocumentRoot /home/Intranet/www.linuxforyou.org
ServerName www.linuxforyou.org
ScriptAlias /cgi-bin/ /home/Intranet/www.linuxforyou.org/cgi-
bin/
ErrorLog /home/Intranet/www.linuxforyou.org/errlog
</VirtualHost>
```

Upload the relevant files to the mentioned location and make sure that the files have umask of 022—that is, others should have 'read and access' Unix rights. There are many methods by which you can upload, such as FTP and secure copy. Now you are ready to browse two or multiple sites. Remember that things can seriously go wrong if your DNS is mis-configured.

The second way is unconventional but followed in most of the intranets and simple to implement. In this method, we map a virtual IP instead of a name to the Apache server. While getting fixed IP-based Internet connectivity, you also need four to five IP addresses along with it. These IP addresses can be used, or you can use internal local IP addresses also, if only some portion of the intranet is to be accessed internally. This is generally preferred because it is easy to implement. Also, you need not bother about DNS and the lack of global IP addresses. The Web browser can approach this site via an IP address.

This first step is to assign a virtual IP address to your Ethernet card. The choice of the Ethernet card is dependent on the requirement—on whether the need is for global access or local access only. As we choose *Eth0* for global and *Eth1* for local IP, we can make the aliases accordingly. Any network tools such as netconf or netcfg can be used to define

virtual IP.

In case of netconf, choose IP aliases for virtual hosts→ eth0→

And define the global virtual IP here along with the subnet mask

Reload your network daemon by

```
/etc/rc.d/init.d/network reload
```

ifconfig should display the virtual IP address, as:

```
eth0      Link encap:Ethernet  HWaddr 00:04:AC:77:29:88
          inet addr:210.210.85.98  Bcast:210.210.85.103
          Mask:255.255.255.248
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:111659 errors:0 dropped:0 overruns:0
frame:0
          TX packets:111290 errors:0 dropped:0 overruns:0
carrier:0
          collisions:1424 txqueuelen:100
          Interrupt:14 Base address:0x74e0

eth0:0    Link encap:Ethernet  HWaddr 00:04:AC:77:29:88
          inet addr:210.210.85.99  Bcast:210.210.85.103
          Mask:255.255.255.248
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:14 Base address:0x74e0
```

Once virtual IP is mapped, you are ready to map it on to the Web server and host the virtual site. The following line (with required changes) will have to be added:

```
<VirtualHost 210.210.85.99>
DocumentRoot /home/iss
ServerName iss
ScriptAlias /cgi-bin/ /home/iss/cgi-bin/
ErrorLog /home/iss/errlog
</VirtualHost>
```

Once you have uploaded the required files or the website to *home/iss*, you should be able to browse the site mentioning the IP address as <http://210.210.85.99>

SECURING THE INTRANET CONTENT

There are two ways of restricting access to documents: either by the hostname of the browser being used, or by asking for a username and password. The former can be used to, for example, restrict documents to be used within a company. However, in the real scenario, people who are accessing the documents are widely dispersed. A mechanism that assigns usernames and passwords before allowing access to a document can be implemented here. This is called user authentication.

Setting up user authentication takes two steps: first, you create a file containing usernames and passwords. Second, you tell the server what resources are to be protected and which general users are allowed (after entering a valid password) to access them.

A list of users and passwords needs to be created in a file. For security reasons, this file should *not* be under the document root. Our example will assume that we create file

users in `/usr/local/etc/` directory (you can choose any but not the document root). Apache has a compiled program called `htpasswd`, which can create the required file with encrypted passwords.

```
htpasswd -c /usr/local/etc/users biswajit
```

The `-c` argument tells `htpasswd` to create a new users file. When you run this command, you will be prompted to enter a password for `biswajit`, and confirm it by entering it again. Other users can be added to the existing file in the same way, except that the `-c` argument is not needed. The same command can also be used to modify the password of an existing user.

After adding a few users, the `/usr/local/etc/users` file might look like this:

```
biswajit:WrU808BHQai36
deepa:iABCQFQs40E8M
mukul:FAdHN3W753sSU
```

The first field is the username, and the second field is the encrypted password.

Now comes the Apache server's configuration part. Look for `AllowOverride None` in your `httpd.conf` file. And replace it with `AllowOverride AuthConfig`. If you want to protect the document root itself, then create a file `htaccess` in the top directory path like

```
AuthName "Only Valid Access"
AuthType Basic
AuthUserFile /usr/local/etc/users
require valid-user
```

The first directive, `AuthName`, specifies a **realm** name for this protection. Once a user has entered a valid username and password, any other resource within the same realm name can be accessed with the same username and password. This can be used to create two areas, which share the same username and password.

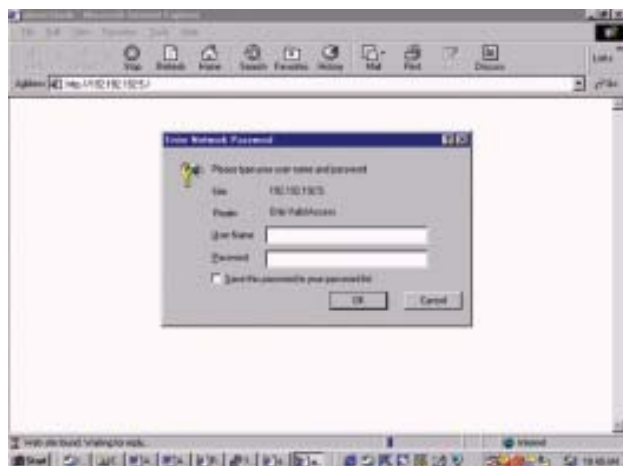


Figure 3

The `AuthType` directive tells the server what protocol is to be used for authentication. At the moment, **Basic** is the only method available. However, a new method, **Digest**, is about to be standardised, and once browsers start to implement it, Digest authentication will provide more security than the Basic authentication.

`AuthUserFile` tells the server the location of the user file created by `htpasswd`. A similar directive, `AuthGroupFile`, can be used to tell the server the location of a group's file.

When you try to access the protected site or the directory via the Web browser, you will be asked for authentication first, as shown in Figure 3.

MAIL SERVER ASPECT OF THE INTRANET

We had a detailed article discussing the implementation of Q-mail based e-mail server in the March issue of 'LINUX For You'. To continue with it, we will discuss a mail server based on **Sendmail** (in future issues).

INSTANT MESSAGING SERVER IMPLEMENTATION

Corporate instant messaging is becoming more and more popular. As a part of intranet implementation, IM (Instant Messaging) cannot remain untouched. The concept of IM is not new to corporate users. It's very frequently and effectively used and is now part of corporate culture. Pioneers of instant messaging are yahoo.com, MSN, and ICQ. Here, let's build our own instant messaging server. There are two technologies for instant messaging:

1. Jabber server
2. Sip-based technology (merging with VoIP)

JABBER SERVER

Jabber is an open XML protocol for realtime exchange of messages and presence between any two points on the Internet. The first application of Jabber technology is an asynchronous, extensible instant messaging platform, and an IM network that offers functionality similar to legacy IM systems such as AIM, ICQ, MSN, and Yahoo. However, Jabber offers several advantages over legacy IM systems.

Let's try to build up the server. Download jabber rpm from `ftp://ftp.scenespot.org/rpms/redhat-7.2/i386/jabber-1.4.2-1.i386.rpm` or from CD.

This version will run on Redhat 7.2 and above. If you don't have this version, download the source from `http://jabberd.jabberstudio.org/downloads/jabber-1.4.2.tar.gz` or get it from the CD.

```
In case of RPM package
#rpm -ivh jabber-1.4.2-1.i386.rpm
Edit /etc/jabber/jabber.xml and modify the line
<host><jabberd:cmdline flag="h">intranet.linuxforyou.com</
jabberd:cmdline></host>
to suit the configuration.
Start the server by
/etc/rc.d/init.d/jabberd start
```

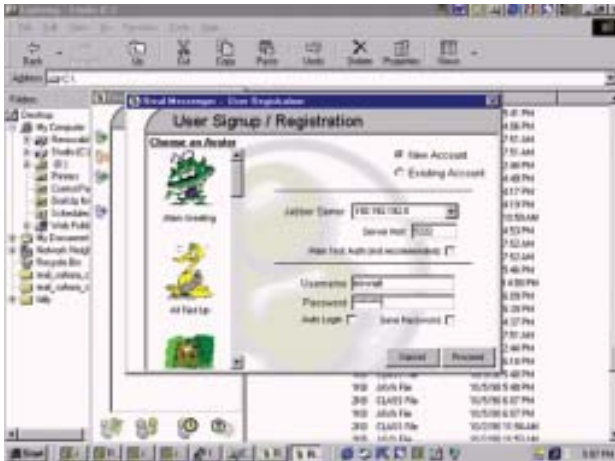


Figure 4



Figure 5

Your server is now running on port 5222
 In case of source download, copy the source in `/usr/local` directory and untar it

```
#tar xvfz jabber-1.4.2.tar.gz
#cd jabber-1.4.2.tar.gz
To build, run "./configure"
and then type "make"
Optional: edit jabber.xml,
edit the <host> section near
the top to replace "localhost"
(or just use the -h flag
below), edit other values as
desired
To start:
./jabberd/jabberd -h
your.domain.name &
```

Once the server is ready, let's see how the client

connects to it. There are a large number of jabber clients available (you can find a list of clients at <http://www.jabber.org/user/clientlist.php>)

We have used and evaluated a rival client for Windows. Download the client from http://rival.chote.net/?set_location=download or from the CD (R3FULL_167.exe) and install it on the client's Windows machine.

The configuration screen is shown in the figure, where you have to choose the Jabber server's IP or the name. Choose username/password and you can now have your own instant messaging.

Advanced users can use the database authentications to protect users and administer them.

SIP SERVER

SIP, the Session Initiation Protocol, is a signalling protocol for Internet conferencing, telephony, presence, events

notification and instant messaging. We will use SIP express router (SER) from <http://iptel.org> to demonstrate the instant messaging solution. This solution can actually go beyond and can act like a VoIP server, where voice can travel to the other IP location.

SER or SIP Express Router is a very fast and flexible SIP (RFC3621) server. Written entirely in C, it can handle thousands of calls per second on low-budget hardware. A C shell-like scripting language provides full control over the server's behaviour. Its modular architecture allows only required functionality to be loaded. The following modules are available: digest authentication, CPL scripts, instant messaging, MySQL support, a presence agent, radius authentication, record routing, an SMS gateway, a Jabber gateway, a transaction module, a registrar, and user location.

You will need Redhat 7.x and above to run this application.

Download the rpm based package from <ftp://ftp.berlios.de/pub/ser/0.8.10/packages/redhat/7.x/ser-0.8.10-2.i386.rpm> or from the CD and install as

```
# rpm -ivh ser-0.8.10-2.i386.rpm
```

SER will start with no modification if you do not require any authentication. Advanced users can go through the documentation to implement authentication based on MySQL, LDAP, or SQL.

```
Start the service by /etc/rc.d/init.d/ser start
```

Once the SIP server starts, you'll have to look for the SIP client. There is a list of SIP clients available at <http://www.iptel.org/info/products/sipphones.php>. But for this implementation, we have used the popular MSN client with

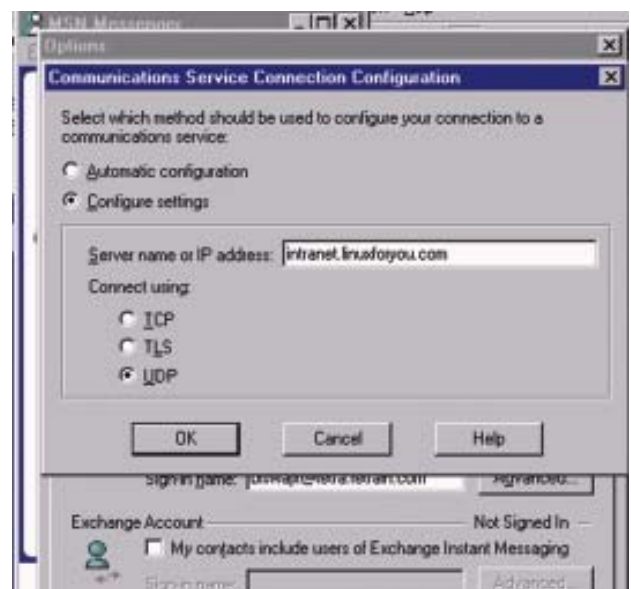


Figure 6

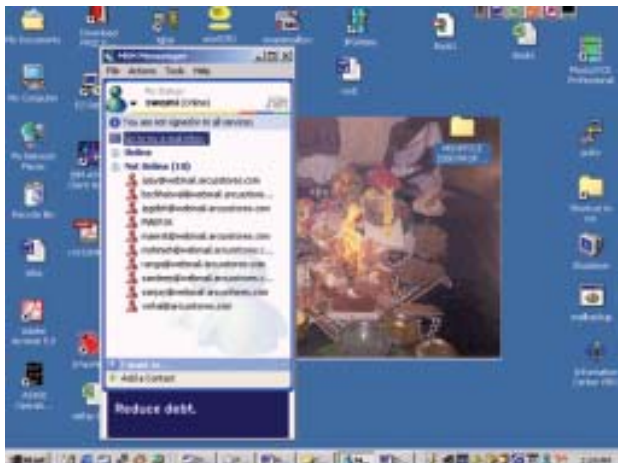


Figure 7

the Exchange patch. Download MSN 4.6 from <http://www.microsoft.com/exchange/downloads/2000/imclient.asp> or from the CD (mmsetup.exe) and install at the client end.

Configure the MSN Client Tools→Options→Accounts →Communication Services

Go to the Advance Tab of Communication Services and configure as shown in the figure, choosing configure settings, server name or the IP address and UDP selected.

Once you sign-in, you will see what is displayed in the Figure 7.

You can have both your favourite Hotmail as well as your corporate login for instant messaging. Make sure port 5060 on which the SIP service is running is open and not blocked by the firewall.

SECURING THIS SERVER VIA A FIREWALL

Firewalls are used for two purposes.

1. To keep people (worms/crackers) out.
2. To keep people (employees/children) in.

There are two types of firewalls.

1. *Filtering firewalls:* These block selected network packets.
2. *Proxy servers* (sometimes called firewalls): These make network connections for you.

Packet filtering is the type of firewall built into the Linux kernel.

Filtering firewalls: A filtering firewall works at the network level. Data is only allowed to leave the system if the firewall rules allow it. As packets arrive, they are filtered by their type, source address, destination address, and port information contained in each packet.

Many network routers have the ability to perform some firewall services. Filtering firewalls can be thought of as a type of router. Because of this, you need a deep understanding of IP packet structure to work with one.

Because very little data is analysed and logged, filtering firewalls take less CPU and create less latency in your network.

Filtering firewalls are more transparent to the user. The user does not have to set up rules in his applications to use the Internet. With most proxy servers, this is not true.

Proxy Servers: Proxies are mostly used to control, or monitor, outbound traffic. Some application proxies cache the requested data. This lowers bandwidth requirements and decreases the access of the same data to the next user. It also gives unquestionable evidence of what was transferred.

There are two types of proxy servers.

1. Application proxies—that do the work for you.
2. SOCKS proxies—that cross wire ports.

The best example is of a person Telnetting to another computer and then Telnetting from there to the outside world. With an application proxy server, the process is automated. As you Telnet to the outside world, the client sends your data to the proxy first. The proxy then connects to the server you requested (the outside world) and returns the data to you.

Because proxy servers are handling all the communications, they can log everything they (or you) do. For HTTP (Web) proxies, this includes every URL you see. For FTP proxies, this includes every file you download. They can even filter out ‘inappropriate’ words from the sites you visit or scan for viruses.

Application proxy servers can authenticate users. Before a connection to the outside world is made, the server can ask the user to login first. To a Web user, this would make every site look like it required a login.

My purpose of including this portion is not to impart knowledge on building firewalls but to give a basic idea about firewalls. Security is a concept that totally depends on the kind of environments you want to build. There cannot be a single script or program to suit the entire requirement. Whenever we want to implement a firewall, we need to have the organisation’s needs under consideration.

A lot can be written on firewalls. Here, I am only providing certain scripts by which you can protect the server to a large extent. Since we have implemented a server with two LAN cards and our server is running the application, we can have a script and run it for basic protection. Redhat Linux also provides certain tools in-built in the installation to build the firewall, which is more of packet filtering.

We have given two firewalls scripts in the CD which you can run to have basic packet filter firewall to suite your requirement (listed under the heading—Script #6).

We hope this article gives you an insight into how you can set up an intranet server for 24 × 7 Internet access. It can help you build a low-cost backbone, on which you can develop intranet applications in a corporate setup. **LFY**

The article is written by Mr Biswajit Banerjee, Director, Tetra Information Services Pvt. Ltd, who are into Linux corporate solutions. He can be contacted at biswajit@tetra.in