Connection Tracking for Legato NetWorker on Linux

David Stes
email: stes@pandora.be

January 3, 2004

Abstract:

This is a ``howto'' on setting up a firewall for a Legato NetWorker server, on Linux. The method that we describe, uses Linux ``iptables'' connection tracking, also known as stateful packet filtering. Instead of defining a *stateless* firewall in terms of TCP ports to be opened, as described in Legato Technical Bulletin 354 ([#!Legato354!#]), we use the Linux module for *RPC connection tracking* for client-initiated Legato backups. For scheduled (server-initiated) backups, we use the Linux module for *RSH connection tracking*. The idea is that the Legato protocols are *remapped* versions of Sun RPC and REXEC (RSH), and that we can therefore use the existing modules for RPC and RSH in order to support Legato. By using connection tracking, we accurately track Legato portmapper requests, and we only *dynamically* open those ports that are being requested by Legato, as they are registered with the Legato portmapper, effectively filtering on specific Legato RPC procedures, instead of filtering on the underlying TCP ports. It is demonstrated that with RPC and RSH connection tracking, Legato backup over a firewall requires *only two ports to be opened*, namely TCP port 7937 (the ``nsrexec'' equivalent of the ``rexec'' port) and TCP/UDP port 7938 (the ``lgtomapper'' equivalent of the ``sunrpc'' port). Of course, those two ports should never be opened to an untrusted network.

Introduction

Connection tracking is a feature of the Linux ``iptables'' firewall (by Rusty Rusell, Harald Welte and others, see [#!netfilter!#]) to *relate* some connections to each other, as being part of a certain *protocol*.

The classic example is FTP connection tracking, where the firewall is FTP-aware, and therefore understands what outbound and inbound *control* and *data* connections are related to each other. Another example is the relation of ICMP control packets to TCP packets, for a certain TCP connection, or simply the relation of all TCP packets of a TCP connection, to the SYN packet that opened the connection.

Just as for FTP, there exist RPC and RSH connection tracking modules, to understand the RPC and RSH protocols, to relate the various TCP connections that are used by these protocols.

Normally, the RPC module, originally written by Marcelo Lima (see [#!Marcelo!#]) tracks connections to the *sunrpc* portmapper (listening on UDP and TCP port 111).

The RSH module, written by Ian Latter (see [#!Latter!#]), normally tracks connections to the *shell* service (listening on TCP port 514). With old versions of the Legato NetWorker software, both Sun RPC and ``rsh'' are being used.

For example, if the Legato NetWorker software is not running on a client, the Legato NetWorker software will fall back to *rsh authentication*, so that the Legato NetWorker server can start scheduled backups on a client, via *rsh*, on old versions of the Legato NetWorker client software.

More recently, with Legato NetWorkerversion 6 or higher, Legato has implemented their own portmapper (listening on TCP port 7938) and Legato also uses *nsrexec authentication* (by making a connection to the ``nsrexecd'' process, usually listening on TCP port 7937) instead of plain *rsh* for savegroup probes.

The idea of this paper is to *remap* the default ports of the Linux firewall modules, and to use these stateful connection tracking modules with port 7938 and 7937, instead of using the default ports of 111 and 514.

We'd like to thank Ian Latter for adding module options, to both RPC and RSH modules, for remapping the default ports for these modules, and for suggesting to use the RSH module for the *nsrexec authentication* as well. Ian Latter also added the options to filter on specific RPC procedures to the RPC module.

Applications of the solution

In this text, we use the *filter* table and the *INPUT* chain of the Linux iptables firewall. This means that in these examples, we filter on *incoming* packets. This is the situation as described, below, of *hardening* a backup server.

In the case of a Linux router, one would apply the rules to the *FORWARD* chain (the packets that are forwarded), instead of the incoming or outgoing packets for the host itself.

Hardening a Linux backup server

A first example of filtering on incoming packets, consists of a firewall running on a Linux backup server (or storage node) itself. Such a Linux backup server, can run iptables on its ethernet interface, and refuse connections from certain clients on a LAN. The Linux backup server can run the full suite of Sun RPC daemons (for NFS etc.), but these services are protected by the firewall, and the firewall only allows incoming connections to the Legato services, for specific backup clients.

Firewall between backup server and DMZ

Another example, is a Linux machine that is configured as a router (firewall) between one or more DMZ networks, and a production LAN. The production LAN can run a Legato NetWorker server on Windows or UNIX, and the Linux router connects the DMZ to the LAN. The Legato NetWorker software is installed on the clients in the DMZ, and the DMZ is backed up by the Legato NetWorker server on the LAN. The Linux firewall tracks incoming connections from the DMZ, towards the Legato NetWorker server on the LAN. The firewall refuses any connection to the Legato NetWorker server, except for those connections that are related to backup and recoveries.

Library sharing with StorageTek ACSLS

It should be possible to use the same technology to open only a single inbound port to support StorageTek ACSLS. This could be used in environments where the amount of control data (indexes) to be transferred between clients in a DMZ and a backup server on a LAN, it too large. In such a situation, it is a good idea to put a dedicated backup server on the DMZ, and use ACSLS library sharing, over the RPC connection tracking firewall.

Installing the software

RedHat Linux

We use two machines, a client machine called *gecko* and a server called *mix*. The machine *mix* is running RedHat Linux and the Legato NetWorker software (7.1 or 6.1.4, the Legato version is of less importance, as long as it is higher than 6.0).

By default, the RPC connection tracking is *not* in the iptables and netfilter firewall packages for RedHat Linux. In any case, in order to use the RPC connection tracking patch, we will need a kernel 2.4.17 or higher, to apply the Netfilter patch on.

RedHat Linux is not really an ideal distribution to compile your own Linux kernel on (I find something like Slackware Linux easier to use for this), but we will use RedHat here, simply because many organizations like to standardize on some of the ``commercially'' supported distributions.

Because RedHat patches their Linux kernel, it is a good idea to install a RedHat source package for the kernel (under /usr/src) because at least this version of the source code must compile. Also, this version comes with a directory ``configs'' with config files for the kernel build process, as distributed by RedHat.

Kernel 2.4.23

It is possible to use the RedHat kernel sources (RedHat 2.4.20-27.7, for example) and to patch it to add the RPC and RSH connection tracking modules. We download a version (must be more recent than 2.4.17 to be able to apply the netfilter RPC connection tracking patch) from:

```
http://www.kernel.org
```

We install this version under /usr/src.

Next, we copy a RedHat config file from the RedHat kernel source package to ``.config''. This (hidden) file is used by the Linux menu configuration for the kernel build process.

Patch-o-matic-20030912

We download a version of NetFilter patch-o-matic from :

```
http://www.netfilter.org
```

This is the kernel part of the RPC connection tracking patch.

The patch must be applied to the 2.4.23 kernel, by running :

```
KERNEL_DIR=/usr/src/linux-2.4.23 ./runme pending
KERNEL_DIR=/usr/src/linux-2.4.23 ./runme extra
```

There are various ``extra'' patches (the one we are interested in is the RPC patch and the RSH patch). We also installed as much of the pending patches as possible, and of the extra patches we only selected the RPC patch (and choosed not to install any of the other contributions to NetFilter).

Important : It is necessary to apply some fixes (patch to the RPC and RSH modules) in order to use the modules with Legato.

The RSH and RPC modules need minor changes to support Legato. These fixes will be in future versions of the RPC and RSH modules. Contact the author of this article for more information.

Building the patched kernel

You should then run ``make menuconfig'' and make sure that the RPC and RSH connection tracking is enabled (because by default it is disabled).

Go into the menu *Networking options*, next go into *Netfilter configuration*, and then press the space bar to compile the *RSH protocol support* and *RPC match support* as modules.

Save the new configuration, and check in the hidden config file,

```
CONFIG_IP_NF_MATCH_RPC=m
CONFIG_IP_NF_RSH=m
```

This option supplies two connection tracking modules; ip_conntrack_rpc_udp and ip_conntrack_rpc_tcp, which track portmapper requests using UDP and TCP respectively.

```
make dep
make bzImage
make modules
make modules_install
```

There was a small problem (undefined symbols) with the modules of the crypto API :

```
 /lib/modules/2.4.23/kernel/crypto
```

This is not a problem, we simply removed that directory and ran ``depmod -a''.

The RPC and RSH modules must be present now, check that the following modules are present :

```
/lib/modules/2.4.23/kernel/net/ipv4/netfilter/ip_conntrack_rpc_tcp.o
/lib/modules/2.4.23/kernel/net/ipv4/netfilter/ip_conntrack_rpc_udp.o
/lib/modules/2.4.23/kernel/net/ipv4/netfilter/ipt_rpc.o
/lib/modules/2.4.23/kernel/net/ipv4/netfilter/ip_conntrack_rsh.o
```

We installed the kernel (bzImage) and Sysmap under /boot, and made a modified version of the RAM disk (initrd).

Finally, we modified the GRUB file and rebooted to the new kernel.

Iptables 1.2.9

We first uninstalled (using rpm -e) the RedHat iptables package.

Then we downloaded version 1.2.9 from :

```
http://www.netfilter.org
```

The iptables package is the user part of the firewall, as opposed to the kernel part (called patch-o-matic).

This should compile easily, except perhaps for bad RedHat header files that may introduce problems (if the wrong files are included).

The following header files may pose problems:

```
ip_conntrack_tuple.h
ip_conntrack.h
```

If you are unlucky, you may have undefined symbols during compilation of iptables, due to these header files from the kernel being incompatible (they are from an old netfilter version).

However, if you use the Linux 2.4.23 kernel and make sure that the header files are from that kernel distribution (not from the RedHat /usr/include/linux, which you can remove and replace by a link to the fresh kernel include files), iptables 1.2.9 will compile.

Configuring the software

Initial verifications

First, to avoid (common) problems of hostname resolution with Legato, we configure and test things without a firewall.

On the client *gecko*, we run the Linux portmapper and Legato NetWorker for Linux (client-only).

```
root@gecko:~# rpcinfo -p gecko
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
390113    1   tcp   7937  nsrexec
```

On the server *mix*, we run the Linux portmapper and Legato NetWorker for Linux (server version). We also run the NFS daemons on our backup server, so the output of ``rpcinfo" is now:

```
[root@mix]# rpcinfo -p mix
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp   1024  status
    100024    1   tcp   1024  status
    391002    2   tcp   1025  sgi_fam
    390113    1   tcp   7937  nsrexec
    390103    2   tcp   9552  nsrd
    390109    2   tcp   9552  nsrstat
    390110    1   tcp   9552  nsrjb
    390103    2   udp   9553  nsrd
    390109    2   udp   9553  nsrstat
    390110    1   udp   9553  nsrjb
    390107    5   tcp   9770  nsrmmdbd
    390107    6   tcp   9770  nsrmmdbd
    390105    5   tcp   9829  nsrindexd
    390105    6   tcp   9829  nsrindexd
    100011    1   udp    935  rquotad
    100011    2   udp    935  rquotad
    100011    1   tcp    938  rquotad
    100011    2   tcp    938  rquotad
    100005    1   udp   1025  mountd
    100005    1   tcp   1213  mountd
    100005    2   udp   1025  mountd
    100005    2   tcp   1213  mountd
    100005    3   udp   1025  mountd
    100005    3   tcp   1213  mountd
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp   1027  nlockmgr
    100021    3   udp   1027  nlockmgr
    100021    4   udp   1027  nlockmgr
    390104  105   tcp   8444  nsrmmd
```

We can do various tests on the client *gecko* and server *mix*.

It is possible to use the ``nwadmin" application on *gecko* and connect to the server *mix*, to see what the server is doing.

As far as the client initiated part of the work concerns, we should be able to make backups and do recoveries on the client :

```
root@!gecko:~# save -s mix /etc
root@!gecko:~# cd /etc;recover -s mix
```

On the server, to test the server initiated actions, we should be able to start a group containing the client *gecko* :

```
[root@mix]# savegrp -c gecko Default
```

Obviously if these actions are not working, then they should be resolved before setting up the firewall.

Another interesting test to do, is to do an NFS mount on the client (assuming that on the server we have set up an /etc/exports file).

```
root@gecko:~# mount -t nfs mix:/ /u
root@gecko:~# umount /u
```

The NFS mount is working without problems because there is no firewall enabled between the server and the client.

We will see later that once RPC connection tracking is enabled, it is possible to filter on RPC procedures, so that it is possible to enable Legato, but to disable NFS.

Setting up modules.conf

Since version 6, Legato uses their own portmapper (implemented as a ``thread" or child process of the ``nsrexec" process). It listens on port 7938.

We should therefore set up the RPC connection tracking to use this port (and perhaps also the standard Sun RPC port of 111, but this is not necessary if all client and servers run Legato 6 or higher).

We add the following lines to /etc/modules.conf :

```
add options ip_conntrack_rsh ports=7937
add options ipt_rpc ports=7938
add options ip_conntrack_rpc_tcp ports=7938
add options ip_conntrack_rpc_udp ports=7938
```

If you have to firewall other RPC services, or if you have very old Legato NetWorker clients running in a DMZ (which you shouldn't be doing anyway) that still uses plain RSH over the firewall, you can do instead :

```
add options ip_conntrack_rsh ports=514,7937
add options ipt_rpc ports=7937,7938
add options ip_conntrack_rpc_tcp ports=111,7938
add options ip_conntrack_rpc_udp ports=111,7938
```

This is the same thing as before, but it will support the default RSH port of 514 and default port of Sun RPC 111, in addition to the *remapped* ports of Legato 6.x. With recent Legato NetWorker software installed on both client and server, this is not required.

Next we run ``depmod -a'' and we test (using modprobe with the verbose option -v) that we can load the RPC connection tracking with the right option :

```
[root@mix]# /sbin/modprobe -v ip_conntrack_rpc_tcp
[root@mix]# /sbin/modprobe -v ip_conntrack_rpc_udp
[root@mix]# /sbin/modprobe -v ipt_rpc
[root@mix]# /sbin/modprobe -v ip_conntrack_rsh
```

Check in the verbose output, that the correct ports option is being used by ``modprobe''.

RedHat Linux using automatic loading of kernel modules, so normally these modules are automatically loaded, and it is not necessary to manually insert the modules in the kernel.

In any case, it is necessary to check (with ``lsmod'' that the RSH is loaded; it will be listed as ``unused'', this is normal, it is being used if it's loaded).

Setting up iptables rules

With the ``iptables'' list option, we can see the current configuration (no firewall) :

```
[root@mix]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Next, we change the policy on the INPUT chain to drop all incoming packets by default (this must be done on the console of the server, of course, since all communication will be halted by this) :

```
[root@mix]# iptables -P INPUT DROP
```

We cannot ping ourselves any longer or do anything else. To change this, we allow all traffic on the loopback interface :

```
[root@mix root]# iptables -A INPUT -j ACCEPT -i lo
```

Next, we open the ``nsrexec'' 7937 port and ``lgtomapper'' 7938 port on the firewall. Note that 7937 is only used over TCP (it is the remapped shell service) and 7938 is both UDP and TCP (this is the remapped portmapper service).

```
iptables -A INPUT -p udp -m state --state NEW -m udp --dport 7937 -j ACCEPT
iptables -A INPUT -p udp -m state --state NEW -m udp --dport 7938 -j ACCEPT
iptables -A INPUT -p tcp -m state --state NEW -m tcp --dport 7938 -j ACCEPT
```

We enable all related ICMP traffic for TCP connections that are being tracked.

```
iptables -A INPUT -m state --state RELATED -j ACCEPT
```

Once the TCP connection is established, we accept incoming packets that are part of a connection that is being tracked :

```
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
```

The above rules have enabled all traffic on the loopback interface, and they have enabled a TCP or UDP connection to the Legato portmapper.

We now configure the RPC connection tracking module, by specifying that Legato RPC procedures must be allowed, regardless of which TCP port that they use. The TCP port is *dynamically* intercepted by the RPC connection tracking module.

```
iptables -t filter -A INPUT -m rpc --rpcs nsrd,nsrmmd,nsrindexd,nsrmmdbd,\\
    nsrstat,nsrjb,rap,rapserv -j ACCEPT
```

The above RPC procedures are from the file /etc/rpc. The numerical values can also be specified, but iptables can look up the values in the /etc/rpc file.

Note that we have not enabled RPC connection tracking to the ``nsrexec'' service; connections to port 7937 are already possible without RPC connection tracking.

The firewall is now configured to drop any incoming traffic, except traffic for backups and recoveries that is under control of the RPC connection tracking module.

It is not necessary, but useful, to ``reject'' some connections instead of silently ignoring them. For example, (see below) if one prefers ``nmap'' to report a ``closed'' state instead of ``filtered'' state for ports on the backup server, one can add a rule :

```
iptables -A INPUT -j REJECT
```

That rule (at the end of the chain) is not really necessary, but can be useful.

The configuration can be saved in a file, so that the firewall can be configured when the server is rebooted.

```
[root@mix]# ./iptables-save > inrpc
[root@mix]# cat inrpc
:INPUT DROP [11:832]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [8297:2170723]
```

```
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 7937 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 7938 -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 7938 -j ACCEPT
-A INPUT -m state --state ESTABLISHED -j ACCEPT
-A INPUT -m state --state RELATED -j ACCEPT
-A INPUT -m rpc --rpcs 390103,390104,390105,390107,390109,390110,
   390101,390102 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-port-unreachable
```

To temporarily remove the firewall, it is sufficient to change the policy :

```
iptables -P INPUT ACCEPT
```

To re-enable the firewall, reset the policy, or restore the entire configuration :

```
iptables -P INPUT DROP
./iptables-restore < inrpc
```

The client *gecko* can now use all of the Legato utilities (save, recover, mminfo, nsradmin etc.) although that the firewall has only a single open port (to 7938).

Scheduled backups

Client-initiated backups work as soon as RPC connection tracking is enabled.

The commands are ran on the client, and connect to the Legato portmapper on the server, and the firewall ``follows'' the subsequent session, until it is closed.

Running scheduled backups however, is different, because these are server-initiated (savegrp) backups, and the server tells the client to do something, the client will ask the server for more information (level, schedule etc.) and finally will start a client-initiated backup.

The server uses a protocol similar to *rexec*, namely *nsrexec*.

Thanks to the *ip_conntrack_rsh* module (listening on the nsrexec port, instead of on the shell port) this also works; the client can open a connection to the server, associated to this connection, for redirecting *stderr* (the *stdin* and *stdout* streams go over the server-initiated connection).

Note that this means that the TCP port 7937 and TCP/UDP 7938 must also be opened on the client, to the server.

Scanning the backup server

Although that many services are running on *mix*, our Linux backup server, if we *scan* the host, and although that it is used for backup and recovery of clients (such as *gecko*), it seems to be nearly invisible if we scan it using ``nmap''.

Nmap is a tool for scanning open ports. Nmap thinks our server is down, even when it is taking backups :

```
root@gecko:# nmap mix

Starting nmap V. 2.54BETA34 ( www.insecure.org/nmap/ )
Note: Host seems down. If it is really up, but blocking our ping probes, try -P0
Nmap run completed -- 1 IP address (0 hosts up) scanned in 30 seconds
```

We have to use the ``no ping'' option since our backup server only supports ICMP as part of the TCP connections for backup and recovery, and doesn't respond to ICMP echo reply requests. Nmap needs a lot of time to scan the host :

```
root@gecko# nmap -P0 -p 7938 mix

Starting nmap V. 2.54BETA34 ( www.insecure.org/nmap/ )
Interesting ports on mix (172.16.0.8):
Port       State       Service
7938/tcp   open        unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

The backup server is listening on many more ports however. For example, the TCP port for the media database service is ``open''

```
[root@mix]# netstat -an | grep 9053
tcp        0      0 0.0.0.0:9053            0.0.0.0:*               LISTEN
udp        0      0 0.0.0.0:9053            0.0.0.0:*

[root@mix]# lsof -i :9053
COMMAND    PID USER   FD   TYPE DEVICE SIZE NODE NAME
nsrd      3022 root    8u  IPv4  46860       UDP *:9053
nsrmmdbd  3027 root   27u  IPv4  46931       TCP *:9053 (LISTEN)
```

On the client *gecko*, thanks to RPC connection tracking, we can use ``mminfo'' which consults the media database service :

```
root@gecko# mminfo -s mix -am
   volume                  written  (%)  expires     read mounts capacity
   mix.001                   18 MB 100%  09/26/04    3 KB     4      0 KB
```

However, when we try to see the open ports of *mix*, nmap is unable to detect that the TCP port 9053 is indeed servicing ``mminfo'' requests from clients.

```
root@gecko# nmap -v -P0 -p 9053 mix

Starting nmap V. 2.54BETA34 ( www.insecure.org/nmap/ )
No tcp,udp, or ICMP scantype specified, assuming vanilla tcp connect() scan. Use -sP if you really don't want to portscan (and ju
Host mix (172.16.0.8) appears to be up ... good.
Initiating Connect() Scan against mix (172.16.0.8)
The Connect() Scan took 36 seconds to scan 1 ports.
Interesting ports on mix (172.16.0.8):
Port       State       Service
9053/tcp   filtered    unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 36 seconds
```

The ``filtered'' state means that nmap kept sending SYN packets, but no reply has been received, so that nmap cannot decide that the port is open or closed, but rather reports the state of the port as ``unknown'' or ``filtered'' (only after trying for about 40 seconds to make a connection).

For UDP scanning, which is even slower, we have :

```
root@gecko# nmap -P0 -sU -p 7938 mix

Starting nmap V. 2.54BETA34 ( www.insecure.org/nmap/ )
Interesting ports on mix (172.16.0.8):
Port       State       Service
7938/udp   open        unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 12 seconds
```

This corresponds to the one open port that accepts connections : the Legato portmapper port; all other ports are protected by the RPC connection tracking.

Because our firewall is filtering on RPC procedures, if we try to access some other RPC based service, such as NFS (as we did in our initial configuration test) the mount will just hang :

```
root@gecko:~# mount -t nfs mix:/ /u
```

Opening the Sun RPC portmapper port will not change this, but if we add an additional rule to the firewall to accept the NFS RPC procedures (using the appropriate -rpcs option), we can also support NFS on the backup server.

Conclusion : backups and security

System engineers sometimes see backups and security as a ``dilemma'' : a centralized backup system like Legato, may be seen as a security risk, since an attacker may obtain all data, of all servers, once the backup server is cracked.

On the other hand, a centralized and well-managed backup system greatly improves security, because when a certain client is compromised, system management can quickly compare modified files with old versions of the files (stored on the Legato NetWorker server).

When there is no easy-to-use, centralized backup system, security may actually decrease, since system engineers cannot easily see whether suspicious binaries were only recently changed.

Assuming that the ``pros'' of a centralized and consolidated backup environment, outweigh the ``cons'' of such backup software, the technical details of firewall configuration are only a minor, technical, issue.

As described in [#!Legato354!#] the following could very well suffice to make backups and recoveries work over a Linux firewall :

```
iptables -t filter -P INPUT DROP
iptables -t filter -A INPUT  -j ACCEPT --protocol tcp --dport 111
iptables -t filter -A INPUT  -j ACCEPT --protocol tcp --dport 600:1023
iptables -t filter -A INPUT  -j ACCEPT --protocol tcp --dport 7937:9936
iptables -t filter -A INPUT  -j ACCEPT --protocol tcp --dport 10001:300
```

But the *dynamic* approach, using RPC and RSH connection tracking, has the advantage over static filtering (as above) that if Legato NetWorker is restarted, and when it allocates new TCP ports to listen on, the firewall automatically and dynamically adapts to the new situation.

And in any case, by simplifying the backup and restore management, it is possible that system administrators can find more time for systems monitoring and intrusion detection, as opposed to wasting time on checking that backups ran fine on a variety of machines in various DMZ networks, running a wide range of different backup softwares.

David Stes
2004-01-03