USENIX Association

Proceedings of the 17th Large Installation Systems Administration Conference

San Diego, CA, USA
October 26–31, 2003

CDSS: Secure Distribution of Software Installation Media Images in a Heterogeneous Environment

Ted Cabeen – Impulse Internet Services
Job Bogan – Consultant

ABSTRACT

CDSS is a framework for the distribution of software installation media images and their contents over multiple file sharing protocols. The CDSS system provides a unique isolated server instance for every accessing user, even when another instance of that server is already running. CDSS uses the Linux host-based firewall system to transparently redirect inbound connections from each user to his specific server instance. By doing so, multiple users can access the CDSS server over the same protocol on the standard port without requiring any special configuration by the user. Each user can only communicate with the server instance that was started explicitly for him and that has been automatically configured by CDSS to allow access only to the files that he has requested.

CDSS is currently implemented as a collection of web and shell scripts that run on Linux servers that support the IPTables and IPChains firewalling systems. CDSS currently supports image distribution via the following protocols: HTTP, FTP, TFTP, NFS, SMB, and AppleShare IP. CDSS can share any filesystem image file stored on the server as well as the individual contents of those images that the server can loopback-mount.

Introduction

In the last five to ten years, many systems have been written to assist in automating the installation and management of large collections of homogeneous servers administered by a small team of systems administrators. These programs have simplified this complex administration task substantially. However, not all installations enjoy the efficiencies provided by such an infrastructure. Many sites have to deal with large heterogeneous infrastructures where there are many system administrators each controlling a small cluster of machines. In this environment, distributing software in a secure, reliable, efficient and effective manner can be quite difficult. In this paper we present the CD Sharing System (CDSS), a collection of scripts and web interfaces for enabling and simplifying the distribution of software and installation images to heterogeneous clients. First we will discuss common distribution solutions before presenting the server-per-user IP redirection solution used in CDSS. We will also cover the specifics of implementing CDSS over common file distribution protocols provided by UNIX servers. Finally, we will discuss the use of the server-per-user IP redirection technique in other applications and future enhancements planned for CDSS.

A very critical element of a software distribution system is security, particularly with proprietary software under a site or multiple-use license. Software vendors rarely condone the wide-scale distribution of images to non-licensed parties, so the goal of a software distributor is to make the software easily accessible to the licensed parties while denying access to everyone else. When the number of licensed parties is very small (a single group) or very large (freely-distributable software), this is not difficult. In the non-trivial cases, there are a few possible solutions.

One solution is to manually distribute software installation media to each eligible group. Each group can then individually convert the image into a format that is best for them. However, this method does not scale to large numbers of groups, and causes significant duplication of effort both on the part of the distributor and the users.

Another possibility is to place the images and image contents on a central distribution server and provide access to the images over standard protocols. A system like this would necessarily require a substantial cross-protocol authentication and access control system to prevent unauthorized access. Unfortunately some of the most important protocols have no authentication systems at all, and the others that do often do not share a common authentication mechanism. This eliminates some protocols from central administration entirely (e.g., TFTP), and makes managing the other protocols a significant administrative hassle.

CDSS is a framework for software installation media image distribution that provides simplified maintenance, enhanced security, and substantial platform independence. CDSS achieves this by running unique and isolated versions of the sharing software for each and every requesting user. On a server providing images to six users, there may be two web servers, three NFS servers, and six FTP servers all running simultaneously. Even in a situation like this the existence of multiple server instances is transparent to the users, as they are automatically forwarded to their unique server instance using the transparent IP redirection provided by the Linux IPTables and IPChains firewall systems. By running a separate server instance for every user, we can disable the built-in authentication systems of each protocol entirely, simplify the configuration and centralize the authentication within CDSS, all without requiring any special configuration on the client's software. Finally, users request and manage their images from a simple web interface that gathers the information necessary to create the share and runs the sharing script.

The CDSS Solution

At its core, CDSS performs five major tasks:

- Image Selection: Users visit a web page listing all of the available images. They select the specific images they need access to, supply the necessary passwords for those images, and specify the IP addresses that will have access to the images.
- Image Preparation: The images that the user has requested are linked into a directory created for the user, and loop-back mounted into this directory.
- Server Configuration and Execution: The servers necessary to provide the protocols requested by the user are configured to allow access to the user's directory and started.
- Firewall Configuration: Once the servers for each protocol have been started, rules are inserted into the running firewall configuration to redirect packets from the user to their non-standard port for each protocol.
- Security: The scripts also ensure that individual images and directories of images are not shared to to those who do not know the required passwords for those images.

Image Selection

The first step in any sharing session is for the user to make an image request. Each request needs to contain a set of images, any passwords required for those images and the IP addresses that will access the images. In order to simplify this process, CDSS contains a relatively simple web interface that allows the user to easily select the set of images they want to share and the protocols that they need access over. The web interface builds the list of available images at run-time, so there is no special configuration required to share a new image other than to place it into the image repository. The images can also be password protected on an individual or per-directory basis. It is also possible for the server administrator to share a set of images directly, without using the web interface.

Figure 1 shows the client web interface on a small CDSS server. We can see that there are multiple directories that contain images. Each directory and file can have a password or set of passwords associated with it. If a user wants to access several files that require passwords, they can place the passwords in the input box for each directory, or they can place multiple passwords in a single input box. We also request a separate username and password from the user at request time. This username and password is used to identify the user's share and to prevent unauthorized removal of active shares. Along with images and passwords, users must select a set of protocols and specify any additional IP addresses that also need access to the images.

Image Preparation

Once a user has successfully requested a set of images, they must be prepared for his use. Initially, all the images are stored together in a master directory

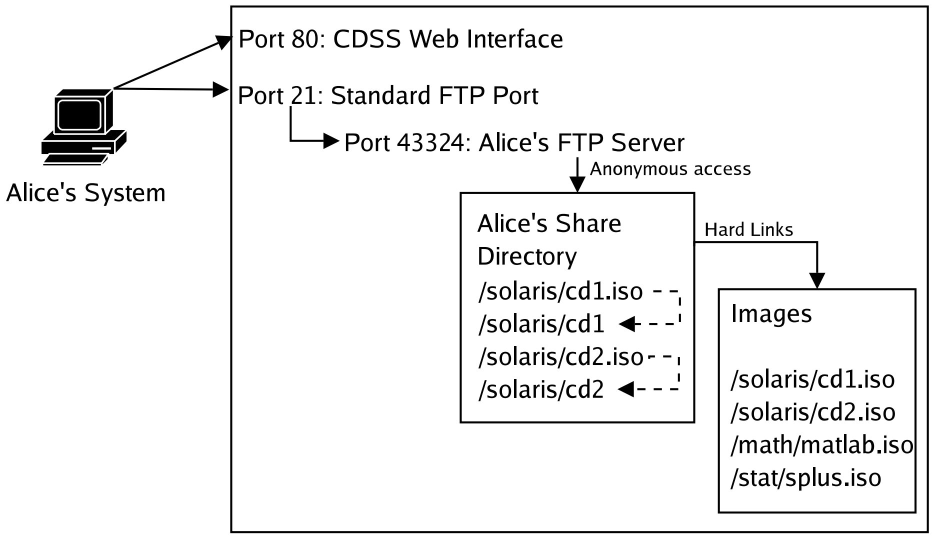

Figure 1: A CDSS server with one client.

tree. When a user makes a request, a new directory is created on the same partition as the master tree and the images requested are hard-linked into the newly-created directory. At this time, we also create individual subdirectories for each image and loop-back mount the images into these directories. This allows the requester to access the contents of each image directly, without having to download the image in its entirety. The loop-back mount can even be used for complete network-based installations directly, without the need to re-share the images. Because the images are loop-back mounted, any CD image that can be mounted by the system can have its contents shared by CDSS. Non-mountable images such as audio CDs can be shared, but only as raw images (the individual tracks or files can't be accessed).

Server Configuration and Execution

After the individual user directory has been created, the servers that provide the requested protocols need to be configured and started. It is important to note that multiple server instances are started for each user, one per protocol. Each server instance is configured to listen for in-bound connections on a non-standard port that is specific to that user. This port is chosen randomly and has no relation to the standard port that the protocol runs on. For each protocol we have a single standard configuration file, if one is required. Before the server instance is started, CDSS copies the standard configuration file into the user's image directory and makes any necessary changes to indicate the directory to be shared and the port to listen on. Then the server is started. Running a separate server instance for each user allows simple automated configuration and enhanced security. However, the redundancy in this system does place an increased load on the server due to the large number of active server instances. CDSS uses lightweight server programs when available and configures them with a minimum of options in order to reduce the CPU and memory usage of the system as a whole.

Firewall Configuration

Once the system has prepared the images and started the necessary servers, the firewall must be configured to allow connections from the requesting user. Connections to the standard ports for each protocol must also be correctly redirected to the unique port allocated to the user for that protocol. Finally, the firewall rules must prevent non-authorized users from connecting directly to the server instances, bypassing the redirect in place on the standard port for each protocol. CDSS currently supports the Linux IPTables and IPChains firewalling systems, although it could be extended to support any firewalling system that can accept rule changes in real-time and transparently redirect connections to alternate ports on the same machine.

Security

Although the firewall configuration described above provides a large measure of security against the possibility of unauthorized access to files, it is also important to secure the web interface effectively. While the web interface is a very convenient way to provide access to large numbers of users without administrator intervention, it also creates the risk of exposing a setuid-root program to the entire allowed user base. CDSS uses extensive checking and verifying of user-provided input both in the web interface and in the master sharing script to prevent the compromise of the hosting server. The CGI web interface itself runs as an unprivileged user and executes the setuid-root script after the user has finalized the request in the web interface. Although it would be ideal to not have to run any part of the system as root at all, root-level permissions are required to maintain the loop-back mounts and adjust the firewall rules. The master sharing script is exceedingly cautious about verifying the data provided by the user and the programs that it executes as root.

For most applications, the CDSS system will need to run on a server that has no unprivileged accounts on it as the media images are stored as world-readable files. It may be possible to use filesystem permissions on the master tree and the individual user directories. However, CDSS does not currently support this functionality. The substantial majority of images shared by CDSS are ISO 9660 images or similar. Images such as these have no access controls inherent in them. However, it is possible to share images that do have filesystem access controls. If this is the case, only those files that are world-readable can be accessed remotely. The loop-back mount system has no ability to edit a mounted image, so we must be careful to avoid images that might be restricted in this way. Such restrictions are also largely ineffective, as the user can always download the complete image and bypass the restrictions at their leisure.

Beyond the security risks inherent in running the CDSS software itself, there is also the possibility of vulnerabilities in the server software for each protocol. In writing CDSS, we have attempted to choose packages with excellent security histories and only the features we need. However, we do extensively rely on the ability of the server software to restrict the users to their assigned directories. Server programs that are unable to restrict access to a single directory tree cannot be used with CDSS. If flaws in the software allow users to break out of their individual directories, then any image file on the system could be accessed and downloaded. In situations where this is a significant concern, additional security could be introduced to the system by chrooting or jailing the server processes into the individual user directories.

Features

Along with the primary tasks listed above, CDSS also has several features that enhance the usefulness of the system, even though they are not required for minimum functionality.

Web Interface

CDSS contains a simple and powerful web interface that allows for individual users to arrange for any number of images to be shared to them with no administrator intervention. The web interface allows for the removal of mounts in order to allow a user to make multiple requests in a single day.

Universal Access

The CDSS system provides enhanced security and efficiency when distributing software installation media images across a large organization. When a user requests an image, they are allowed to specify the IP addresses that are able to access this images being shared. The means that you can share images to machines that are half-way around the world, at home, or even not installed yet. This flexibility is important in larger organizations, where it might be necessary to support remote users without incurring the costs and risks involved in sending CDs to the remote sites, where security and staffing may be a problem. It is also possible to restrict access to the system to IP addresses within the organization to prevent sharing to unauthorized systems.

Securing Any Protocol

CDSS also enhances the security of some protocols that were originally designed with no security mechanisms at all. The most common of these is TFTP, which is primarily used to distribute firmware images to network devices and provide boot-up files to disk-less machines and automated installation systems. Other than blocking access to TFTP entirely, it is impossible for a vanilla TFTP server to provide different sets of files to different clients. By running individual servers for each client, CDSS allows one master server to distribute files over TFTP to many clients without requiring specific or difficult setup on the TFTP client. Other protocols like HTTP and NFS also benefit from this feature, although they do have rarely used authentication systems of their own.

Implementation Details

Although the operational concepts behind CDSS are relatively simple, there are some subtleties in implementing the firewalling and sharing each of the protocols supported by CDSS. We will go over the basics of the program, and then discuss the methods and server software that we chose for each protocol.

Core Operation

The central functionality in CDSS is implemented in a setuid-root perl script called share.pl. Share.pl takes a number of arguments, including the username of the user sharing the files, the IP addresses to be granted access, the files being requested and the passwords for the images. share.pl runs under the perl taint system, so all of the user-provided data must be checked to ensure that it does not contain any malicious strings or data. Once the data has been checked, a directory is created for the user, and the images requested are hard-linked into that directory. Hard-linking the files allows the servers to chroot themselves into the individual user directory without actually copying the files. Hard-links also eliminate any possible problems with access permissions or the inability of sharing software to follow soft-links. However, hard-linking requires that all of the shared files must reside on a single disk partition. CDSS servers commonly use RAID or LVM in order to get drives that are large enough to hold the substantial amounts of data on installation media.

As a convenience to the user, we also loop-back mount the requested images into newly created subdirectories for each image. Although loop-back mounting of images is fully supported by the Linux kernel, it is not enabled in some distributor's default kernels. Even in the kernels where it is supported, the maximum number of simultaneous loop-back mounts is limited to 8. Adjusting this requires modifying the max_loop constant in the drivers/block/loop.c file in the kernel source. Because of this limitation, every CDSS server will need a custom compiled Linux kernel.

The configuration files that are used to start the server instances are not the files that initially come with the software. CDSS has a special repository of custom configuration files that are designed to be automatically configured for the per-user alternate port system used by CDSS. In general, the files themselves undergo few changes for use with CDSS. Once the configuration files are in place, the servers are started and the firewall rules are configured to allow access.

When a user has completed using a share, they are expected to return to the web interface and remove it. However, it's very common for users to neglect to do so. CDSS comes with a shell script intended to be run out of cron that will automatically remove any share that has been in place for more than 24 hours. Shares that are needed for more than 24 hours can be specially configured, but are discouraged. It could be possible to use the automount system to manage the loop-back mounts used by the CDSS system, but the automount systems we looked at aren't well suited to CDSS. The automounter interfaces directly with the kernel, and only a single automount process should be run on a system at a time. Also, most automount systems do not have the ability to make other arbitrary changes to the system when it is time to unmount an active drive, and for CDSS, the unmounting the loop-back is only one of the steps in removing a share.

Firewall Operation

CDSS relies heavily on the IP redirection features that are part of the Linux IPTables and IPChains host-based firewalling systems. Without IP redirection, every user would have to specifically configure their clients to connect with the server on alternate ports. Some lightweight clients for network hardware devices and embedded systems do not have systems to allow for this level of customization. The IP redirection allows the system to support multiple users on the same port simultaneously without requiring any special configuration on the part of the client.

The firewall is configured as follows. A unique chain or table is created for each requesting user, and all of his rules are placed into this chain. With the IP-Tables system two chains have to be created, one in the PREROUTING table, and one in the INPUT table. We then link these newly created tables into the primary tables. All of the rules for a particular user are placed into these tables, so that they can be easily removed. Most protocols only require two firewall rules: one to allow traffic from their IP to the randomly allocated port for this protocol, and one redirecting the user's traffic from the standard port to the randomly allocated one. However, some protocols (such as NFS) require a more involved setup.

Here are the commands run by CDSS to setup to firewall for a HTTP-based share under the IPChains firewall system. The IPTables firewall requires a few more commands because the redirect target is only supported in the nat tables. Figure 2 shows two clients accessing shares and how the IP redirection allows them to simultaneously access two different server instances on the same port.

In this example, most variables are self-explanatory. The actual server for this user resides on $redirport, and $ip is the IP of the client for this image.

```
$IPCHAINS -N $username
$IPCHAINS -I $SYSTEM_CHAIN \
   -i $INTERFACE -j $username
$IPCHAINS -I $username \
   -p $proto -s $ip -d $MYIP \
   $dport -j REDIRECT $redirport
$IPCHAINS -I $username \
   -p $proto -s $ip -d $MYIP \
   $dports -j ACCEPT
```

When it is time to remove a share, the removal script flushes and removes the entire chain created for that user and the rule in the system chain that activates the user's chain.

Protocol Specifics

Currently, CDSS supports file access over the following protocols: HTTP, FTP, TFTP, NFS, SMB, and AppleShare IP. In this section, we will look at each protocol and discuss the server software used and any non-standard configuration that is necessary.

HTTP

As a protocol with near-universal support, HTTP can provide quick, convenient access to almost any client. We chose Boa [5] as the server for the HTTP protocol as it is very lightweight, easy to configure, and high-performance. The only configuration options that need to be set are the port and the directory to be shared, and no special firewalling rules are necessary. Since the client web interface runs on the standard web port, we provide HTTP access on port 8080, the standard secondary port for HTTP. Since this is a non-standard port, we include a clickable link in the output of the sharing script. Since execution over HTTP is not commonly supported, HTTP is most often used to get a single file or files out of a large image without having to download the complete image. Boa also provides internal support for HTTP continue and internal index generation.

TFTP

TFTP is a key protocol for CDSS to support, both due to its extensive use in automated installation and embedded systems, and because it lacks any authentication system at all. By providing TFTP through the CDSS system, a layer of security can be placed in front of this normally insecure program. In many environments, the security issues of TFTP are

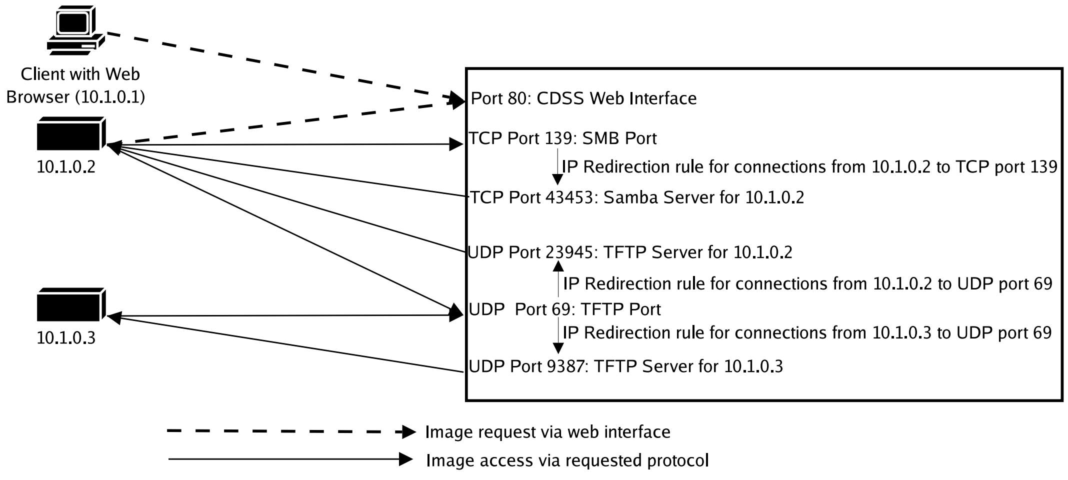
Client with Web Browser (10.1.0.1)
10.1.0.2
10.1.0.3
Port 80: CDSS Web Interface
TCP Port 139: SMB Port
IP Redirection rule for connections from 10.1.0.2 to TCP port 139
TCP Port 43453: Samba Server for 10.1.0.2
UDP Port 23945: TFTP Server for 10.1.0.2
IP Redirection rule for connections from 10.1.0.2 to UDP port 69
UDP Port 69: TFTP Port
IP Redirection rule for connections from 10.1.0.3 to UDP port 69
UDP Port 9387: TFTP Server for 10.1.0.3
Image request via web interface
Image access via requested protocol

Figure 2: Multiple clients access a CDSS system.

ameliorated by firewalling the TFTP port to the small set of IPs that the TFTP clients reside on. CDSS allows for a central distributor to maintain a TFTP server without the overhead of maintaining a lengthy firewall access list.

TFTP is traditionally run out of inetd, running on a UDP port. In a CDSS environment, we use netcat compiled with the -DGAPING_SECURITY_HOLE. That allows the use of the -e argument, which emulates inetd for a single program. Because netcat only allows a program name to be provided with the -e argument, we also create a simple shell script that contains the arguments to tftpd. Other than the script and the standard firewall rules, no special configuration is needed for tftp.

AppleShare IP

AppleShare IP is a recent addition to the Apple file sharing protocol. Before its release in MacOS 8, AppleShare was only available over the AppleTalk protocol. However, we can now use AppleShare IP to provide access to the images on the CDSS system to older Macintosh systems that are not running MacOS X. (MacOS X supports Windows sharing as well.) For MacOS systems prior to MacOS 8, the only supported option is to copy the files over using HTTP or FTP. We use the netatalk package for the actual sharing, and no special firewall rules are required beyond the single redirect.

SMB

File sharing for Windows systems is provided by the SMB protocol. Although it is quite a complex protocol, we only need to provide the sharing portion of it. The most difficult part of sharing over SMB is actually client support rather than server support. The SAMBA package has excellent support for SMB and is easy to configure, but many windows machines have difficulty accessing IP-based shares. However, with some careful configuration, most windows machines can access IP-based SMB shares.

Configuring Windows clients to access IP-based SMB shares can be quite difficult. Traditionally, Windows has used an Ethernet broadcast protocol known as NetBIOS to enable file sharing between windows machines. When IP based sharing was introduced in windows NT, the WINS system was created to create linkages between NetBIOS names and IP addresses. Unfortunately, the WINS and NetBIOS systems often cause difficulties in accessing IP shares. We have had the most success with Windows NT, Windows 2000, and Windows XP machines, although Windows 95, Windows 98, and Windows ME machines can attach to IP-based shares if NetBIOS is completely disabled and the hosts are correctly listed in DNS or the Windows LMHOSTS file, which stores linkages between machine names and IP addresses.

NFS

Of all the protocols that we support, NFS requires the most complex configuration. On most systems, NFS support is provided by the kernel itself. However, since CDSS requires multiple running copies of the same server, we have to use the user space NFS server. The user space NFS server consists of multiple small programs that each support one part of the NFS protocol. CDSS requires only nfsd and mountd. An interesting element of NFS is that is traditionally runs on randomly allocated ports, unlike all of the other protocols listed here. In order to direct clients to the proper port, the nfs daemon registers itself with the RPC portmapper. When clients want to mount an nfs share, they discover the actual port for NFS from the portmapper, and then connect directly.

Because we are using IP redirection to move the connections to a different port for every user, we have to override the redirections made automatically by the portmapper. This is performed by replacing the portmap configuration after each NFS server is run. By replacing the configuration with a standard one, we can then configure our IP redirection system to correctly send each client to their individual NFS servers. This sort of configuration replacement is necessary for any protocol that uses the RPC system to manage its communication. Unfortunately, some systems do not have a user space NFS server, or they do not allow for the NFS server to open a specific port. Non-standard NFS systems or modifications to the Linux user space NFS server may be required in these cases. Because of the transient nature of CDSS shares, we strongly recommend that clients using NFS configure their mounts as soft mounts. This allows the client to kill processes waiting for data from the NFS server or unmount the share entirely if their share has been removed.

Other Applications

Although CDSS was written to share software installation media images, it could easily be extended to provide access to programs or other services. It's important to note that we have endeavored to use lightweight servers and protocols whenever possible. Although it is possible to use the IP redirection techniques of CDSS with more substantial systems, the ability of the system to service multiple users would be hindered. CDSS is best suited to protocols without authentication built-in, or when different users need completely different configuration and the software does not support having per-user settings.

Conclusion

This paper introduces a system for distributing software installation media images to a large set of heterogeneous clients. By using a unique server instance for every user, we are able to simplify and largely automate the configuration of the unique server for each user. Even though we may be running many copies of the same server, per-user IP redirection allows all of the server instances to appear on the standard port for the system, even though they are actually completely separate. This system allows for enhanced security and efficiency while reducing the administrative overhead of maintaining such a server.

Availability

CDSS is licensed under the GPL and is currently being hosted at SourceForge. The master web site is http://cdss.sf.net/ .

Author Information

Ted Cabeen has been working with UNIX systems for 10 years, and has been administrating them professionally for the last six. He developed CDSS at the University of Chicago and now works for Impulse Internet Services in Santa Barbara, CA. He can be reached at secabeen@pobox.com .

Job Bogan is currently working as a consultant specializing in clustering and administration for large academic institutions and companies.

References

[1] SMCC, ‘‘LOFS: Loopback Virtual File System,’’ *SunOS 5.5.1 Reference Manual*, Section 7, Sun Microsystems, 20 March, 1992.

[2] *Linux IP Firewalling Chains*, http://www.netfilter.org/ipchains/ .

[3] *The Netfilter/IPtables Project*, http://www.netfilter.org/ .

[4] Olaf Kirch, *Universal NFS Server for Linux*, ftp://linux.mathematik.tu-darmstadt.de/pub/linux/people/okir/ .

[5] *The Boa Webserver*, http://www.boa.org .

[6] *Netatalk, A Kernel Level Implementation of the AppleTalk Protocol Suite*, http://netatalk.sf.net/ .

[7] *Samba, A SMB/CIFS Server Suite*, http://www.samba.org/ .

[8] *ProFTPD, Highly Configurable GPL-Licensed FTP server Softwarek*, http://www.proftpd.org/ .