

PXE-Installations-Server unter Linux

**DOKUMENTATION DES IHK-PROJEKTES ZUR
ABSCHLUSSPRÜFUNG ZUM FACHINFORMATIKER
SYSTEMINTEGRATION**

VON MALTE EISMANN

Zusammenfassung

Server zur automatischen PXE-Installation der RedHat-Linux-Server im Rechenzentrum der Colt-Telecom GmbH in Berlin-Moabit

Inhaltsverzeichnis

1 Firmenprofil	3
1.1 Projektumfeld	3
1.2 Schnittstellendefinition	3
1.3 Zeitplanung des Projektes	3
2 Vorbetrachtung	4
2.1 Ist-Analyse	4
2.2 Soll-Konzept	4
2.3 Fachgespräche	4
2.4 Wirtschaftliche Betrachtung des Projektes	4
3 Analyse	4
3.1 Was macht eigentlich...	4
3.1.1 die Installations-Konfigurationsdatei <i>ks.cfg</i>	4
3.1.2 das Pre-Execution-Environment	5
3.2 Verschiedene Kickstartvarianten im Vergleich	5
3.2.1 Kickstart mit Diskette und CD/DVD	5
3.2.2 Kickstart mit Disketten und FTP/NFS-Server	5
3.2.3 Kickstart mit PXE-Boot	5
3.2.4 Fazit	5
3.3 Erweiterbarkeit des Installationservers	6
4 Realisierung	6
4.1 Aufbau der Testumgebung	6
4.2 Inbetriebnahme des Installationservers	6
4.3 Installation DHCP-Server	6
4.4 Installation FTP-Server	7
4.5 NFS-Server Freigabe	7
4.6 Installation TFTP-Server	7
4.7 Bootloader bereitlegen	7
4.8 Installationsquellen aktualisieren	7
5 Logging	8
5.1 ...der Installationen	8
5.2 ...des Paketupdates auf dem Server	8
5.3 ...der FTP-Serverzugriffe	8
6 Härten des Systems	8
6.1 Generelles	8
6.2 mehrere Netzwerkkarten-mehrere Netze	8
6.3 Firewall	9
6.4 Benutzer	9
6.5 überflüssige Pakete entfernen	9
7 aufgetretene Probleme	9
7.1 Bootloader	9
7.2 bei der <i>ks.cfg</i>	9
7.3 Serverhardware	10
8 Rollout	10
8.1 Mitarbeiterinweisung	10
9 Qualitätssicherung	10
10 Soll-Ist-Vergleich	10

11 Erklärung	11
A Anhang	11
A.1 Glossar	11
A.2 Literaturquellen	11
A.3 Konfigurationsdateien	12
A.3.1 ks.cfg	12
A.3.2 dhcpd.conf	18
A.3.3 vsftpd.conf	19
A.3.4 fstab	20
A.3.5 /etc/exports	20
A.3.6 /etc/xinetd.d/tftp	20
A.3.7 /etc/mirror.defaults	20
A.3.8 /etc/logrotate.conf	21
A.4 Bootloader und Skripte	22
A.4.1 Installations-PXE-Bootloader-Konfiguration	22
A.4.2 default PXE-Bootloader-Konfiguration	22
A.4.3 PXEcfg_changer.pl	22
A.4.4 gethostip.c	23
A.4.5 update.sh	25
A.5 Logfiles	26
A.5.1 Auszug Installation /root/install.log	26
A.5.2 Auszug .mirror-Logfile	27

1 Firmenprofil

Seit 1992 ist die Colt Telecom (COLT=City of London Telecom) ein europaweit tätiges Unternehmen, das mit 2500 Mitarbeitern Geschäftskunden betreut. Hauptgeschäft ist das Colt-eigene Glasfasernetz, das zwischen wichtigen europäischen Metropolen verlegt wurde. Die Colt Telecom bietet ihren Kunden Telekommunikationsdienste: Sprache, Daten, Internet. Die Colt Telecom Deutschland GmbH ist eine von neun europäischen Tochtergesellschaften, mit Firmensitz in Frankfurt/Main. In Berlin betreibt die Colt Telecom GmbH ein Rechenzentrum, das Geschäftskunden die nötige Infrastruktur für ihre IT-Umgebung bietet, sowie eine Verwaltungszentrale für den norddeutschen Raum.

1.1 Projektumfeld

Das Praktikum fand im Rechenzentrum in Berlin-Moabit statt, dort arbeiten zur Zeit 15 Angestellte, unterteilt in die Teamgruppen Unix, Windows, Networking und Infrastruktur. In der Abteilung Unix arbeiten zwei Mitglieder eines europaweit tätigen Teams, des Unix-Teams mit mehr als vierzig Mitarbeitern. Durch die straffe Organisation in fachlichen Teams kann die Colt Telecom ihren Kunden einen 24/7 Support bieten, sowie gleichzeitige Neuimplementationen von Servern durchführen.

Ein Projekt ist in diesem Zusammenhang anzuführen, das während meiner Zeit bei der Colt Telecom durchgeführt wurde: die Webserver-Cluster für die nun anstehende Europawahl. Die geclusterten Server werden wegen besserer Redundanz in mehreren Rechenzentren Europas verteilt, um den interessierten Benutzern gute Erreichbarkeit zu gewährleisten. Während der Wahl wird die Main-Database eines Server vom Kunden upgedated (aktuelle Wahlergebnisse), und per Perl-Skript an die anderen Webserver übertragen.

1.2 Schnittstellendefinition

Das Projekt ist mit Herrn Tumm, ebenfalls Praktikant im Unix-Team bei der Colt Telecom GmbH, durchgeführt worden. Die Schnittstelle ist ganz klar definiert, Herr Tumm hat das Webinterface auf Basis von Perl entwickelt, ich habe die gesamte Implementation des Servers durchgeführt. Herr Tumm kann excellent programmieren, er war somit ein guter Partner für dieses Projekt. Die Administratoren sahen in der Konstellation einen guten Zusammenschluss.

1.3 Zeitplanung des Projektes

Zur Erstellung eines zeitlichen Ablaufplanes des Projektes werden Aufgaben definiert und nach Aufwand eingeschätzt. Der nachfolgenden Tabelle können Sie die tatsächlich aufgetretenen Zeiten der einzelnen Arbeitsschritte entnehmen.

Projektphase	Arbeitsschritt	Aufwand in Stunden
Projektvorbereitung	Soll-/Istvergleich	2
	Analyse Installationsmethoden	4
	Testumgebung definieren	0,5
Realisierung	Recherche der erforderlichen Dienste	1
	Testumgebung aufbauen	1
	Installation/Konfiguration DHCPd	2
	Installation/Konfiguration FTPd	0,5
	Installation/Konfiguration NFSd	0,5
	Installation/Konfiguration TFTPd	0,5
	Bootloader erstellen	3
	Mirror Installationsquellen erstellen	4
	Installation/Konfiguration Firewall	1
	Dokumentation	6,5
Testphase	Dienste des Servers testen	2
	Test-Installation durchführen	0,5
	aufgetretene Probleme beheben	4
Rollout	Abnahme durch Colt-Admin	0,5
	Implementation im Rechenzentrum	2
	Mitarbeiterweisung	1
Summe	19 Schritte	36,5 Stunden

2 Vorbetrachtung

2.1 Ist-Analyse

Zur Zeit werden alle Server von einem FTP-Server installiert, die dazu notwendigen Medien sind eine Start-CD, eine Diskette mit dem Kickstartdatei *ks.cfg*, und der FTP-Server auf dem die notwendigen/upgedateten Pakete zur Verfügung gestellt werden. Der zu installierende Server wird von der CD gebootet, automatisch wird die Kickstartdatei von Diskette eingelesen, und danach vom FTP-Server installiert. Dazu muss ein Mitarbeiter des Rechenzentrums zum Server gehen und die Medien einlegen, bzw. nach der Installation wieder aus den Laufwerken entfernen, um ein erneutes Installieren nach dem Reboot zu vermeiden.

2.2 Soll-Konzept

Ein neuer Server soll installiert werden, indem sein Pre-eXecution-Environment dazu benutzt wird. Der Client soll sich automatisch zum Installationsserver verbinden und die dort für ihn abgelegte *ks.cfg* benutzen, um sich selber zu installieren. Über ein Webinterface wird die neue *ks.cfg* für den Client bereit gestellt, ebenso notwendige Veränderungen in der *dhcpd.conf* werden von einem Skript abgearbeitet.

2.3 Fachgespräche

Zur Durchführung des Projektes wurden mit den zuständigen Administratoren tiefgreifende Fachgespräche geführt, Vieles war noch unklar, gerade Sicherheitsfragen im Rechenzentrum, Netzwerk-Know-How und die Hardware der Server waren neue Fachgebiete. Herr Menz (Solaris) und Herr Sasse (Linux) erklärten ihre Vorstellungen und bestehenden Systeme. Mit Herrn Werner und Frau Wagner (beide Team Network) wurden spezielle, das Rechenzentrum betreffende Netzwerkfragen geklärt, nicht zuletzt bei der Installation der Kundenserver während des Projektes.

2.4 Wirtschaftliche Betrachtung des Projektes

Die wirtschaftliche Komponente des Installations-Servers liegt auf der Hand, jetzt kann sogar Herr Wiebe (Team Server-Infrastruktur) mit wenigen Handgriffen einen neuen Kundenserver einrichten und dem Unix-Team fertig installiert zur Verfügung stellen. Das Unix-Team kann sich somit schnell der Implementation widmen, der Kunde kann seinen bestellten Server sofort in Empfang nehmen. Durch den Installationsserver kommt es zu einer Zeitersparnis von etwa einer Stunde pro Installation, das summiert sich schnell. Ein Administrator kostet für den Kunden mehr als einhundert Euro in der Stunde, dieser Betrag kann als Reingewinn bei Installationsfestpreis betrachtet werden, oder an den Kunden als Rabatt weitergegeben werden um einen marktwirtschaftlichen Vorsprung gegenüber der Konkurrenz zu haben. Bei der Colt Telecom GmbH werden pro Monat etwa 10 Kunden einen neuen Linux-Server bestellen, somit kann pro Monat eine Einsparung von 10 Manhours a 100 Euro kalkuliert werden, das sind 1000 Euro pro Monat.

3 Analyse

3.1 Was macht eigentlich...

3.1.1 die Installations-Konfigurationsdatei *ks.cfg*

In der *ks.cfg* werden die Parameter festgelegt, die bei einer normalen Installation interaktiv vom Benutzer abgefragt werden. Hier wird sowohl die Partitionierung der Festplatte beschrieben, als auch die Liste zu installierender Pakete spezifiziert. Des Weiteren lassen sich Skripte angeben, die unmittelbar vor bzw. nach der Installation auszuführen sind. Ist die Datei erstellt, installiert sich der Rechner automatisch, er benötigt keine weiteren Informationen. Auf Einhaltung einer bestimmten Reihenfolge der Einträge in der Datei ist dringend zu achten:

- Angabe der Schlüsselworte, wobei die zugehörigen Parameter in einer Zeile stehen müssen (die Reihenfolge oder Anordnung der Schlüsselworte untereinander spielt keine Rolle)
- Liste der zu installierenden Pakete oder Serien; eine Beschreibung befindet sich auf der CDROM unter */Red-Hat/base/comps*
- Optionales Pre- und Post-Install-Skript

Eine typische Anwendung des Preinstall-Skripts ist das Anlegen einer Sicherungskopie der durch den Installationsvorgang zu überschreibenden Partitionen. In einem Postinstall-Skript ließen sich bspw. erste Benutzer im System einrichten.

Im Anhang finden Sie ein Beispiel (A.3.1), weiterführende Informationen auf der RedHat-Seite <http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/custom-guide/sl-kickstart2-file.html>.

3.1.2 das Pre-Execution-Environment

PXE bedeutet *pre-boot execution environment* und ist ein spezielles Boot-Image für Netzwerkkarten. Die Software auf der Karte ist standardisiert und nicht mehr projektspezifisch, wie z.B. die Software für Etherboot. Für die PXE-Unterstützung unter Linux sorgt das Syslinux-Projekt bzw. sein Ableger PXELinux. Sofern eine PXE-fähige Netzwerkkarte zur Verfügung steht wird es leicht, einen Rechner aus dem Netz zu booten. Im Prinzip läßt sich damit jedes System starten, welches auch per Boot-Diskette zu starten wäre.

Umfangreiche Informationen zum Netboot-Prozess bekommen Sie aus dem Developer-Manual des Etherboot-Projektes unter <http://etherboot.sourceforge.net/doc/html/devman/extension.html>

3.2 Verschiedene Kickstartvarianten im Vergleich

3.2.1 Kickstart mit Diskette und CD/DVD

Bei dieser Variante wird dem Client eine Diskette mit der *ks.cfg* präsentiert, die Installation erfolgt von Distributions-CD's. Nach erfolgreichem Reboot muss eine Aktualisierung des Systems vorgenommen werden, um relevante Sicherheitspatches und -updates einzuspielen.

3.2.2 Kickstart mit Disketten und FTP/NFS-Server

In dieser etwas fortschrittlicheren Herangehensweise wird die Installation mit bereits upgedateten Quellen durchgeführt. Drei Disketten sind zur Installation eines Clients notwendig:

- Bootdiskette mit kernel und initrd (ramdisk)
- Treiberdiskette zur Unterstützung von Netzwerkkarten
- Treiberdiskette zur Unterstützung von SCSI- bzw. RAID-Controllern

Der Client wird von den Disketten gebootet, findet die *ks.cfg* auf der Bootdiskette, und installiert vom zentralen FTP- bzw. NFS-Server, dazu kann auch der aktuelle Server FTP-Server von RedHat <ftp://ftp.redhat.com> im Internet benutzt werden.

3.2.3 Kickstart mit PXE-Boot

Beim umfangreichen und individuellen PXE-Kickstart bootet der Client vom Boot-PROM der Netzwerkkarte, dabei erhält er ein minimales System um eine Verbindung zu einem DHCP-Server aufzubauen. Dieser teilt dem Client IP-Adresse und zu bootendes Kernelimage mit. Via TFTP erhält der Client jetzt das Kernel-Image und bootet den Linux-Kernel. In dem Kernel-Image sind bereits Treiber für NICs und Storage-Controller enthalten. Des Weiteren findet der Client dort die *ks.cfg*, die für ihn auf dem NFS des Installationservers bereit gelegt wurde.

3.2.4 Fazit

Für die meisten Installationen eines Servers würde der Aufwand einen PXE-Boot-Server zu installieren den Nutzen nicht rechtfertigen. In Anbetracht der Tatsache, dass bei der COLT Telekom täglich Neu-Installationen durchgeführt werden, rechnet sich die Anschaffung eines PXE-Installationservers. Dadurch kommt eine Einsparung von etwa einer Man-Hour pro Installation zustande.

Art der Installation	Zeitaufwand	Voraussetzung
mit Diskette und CD/DVD	1 Stunde	CD/DVD-ROM und Floppy
mit Disketten und FTP/NFS-Server	30 Minuten	CD-ROM und Floppy, FTP/NFS-Server
mit PXE-Boot	5-10 Minuten	PXE fähige NIC, PXE-Installationsserver

3.3 Erweiterbarkeit des Installationservers

Im Rechenzentrum kommt es immer wieder vor, dass ein Disaster-Recovery-System gebootet werden muss. Das kann auch der Installationsserver übernehmen. Die Disaster-Recovery-Variante ist nur zu administrativen Zwecken zu gebrauchen. Dabei wird ein alternatives Linux-System mit voller Funktionalität in den Speicher des Clients nach Knoppix-Vorbild geladen. Der Administrator erhält so SSH-Zugriff auf einen Server der normalerweise nicht mehr zu erreichen ist. Damit stehen ihm verschiedenste Möglichkeiten zum Disaster-Recovery in Notsituationen zur Verfügung. In diesem Zusammenhang möchte ich das Zurückschreiben eines zuvor vorgenommenen Backups anführen.

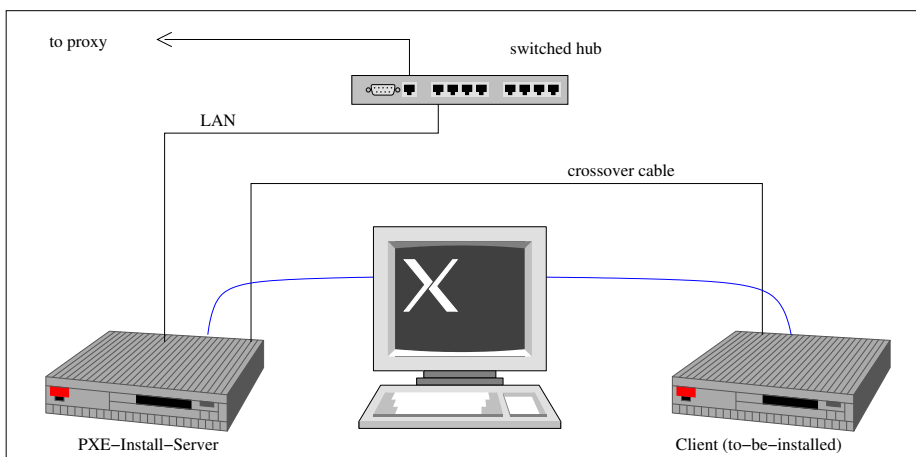
Weiterhin eignet sich der Installationsserver als FTP-Proxy für das Rechenzentrum, er hält die aktuellen Quellen und ebenfalls zuvor auf Kundenmaschinen installierte Pakete. Hier ist beispielsweise der Webserver Apache anzuführen, bei dem es wichtig ist, die richtige Version zu installieren.

Zusätzlich besteht die Möglichkeit, mit geringem Mehraufwand diverse Distributionen mit dem Installationsserver zu betreuen.

4 Realisierung

4.1 Aufbau der Testumgebung

Zum Einsatz kommen hier zwei Rennix-Server 1HE mit Raid1 (je 2mal 18Gb), 1Ghz Intel-CPU, zwei Fast-Ethernet on-board NIC's, PXE-bootfähig. Die beiden Maschinen sind über ein Twisted-Pair Crossover-Kabel miteinander verbunden und hängen im Testlab der Colt Telekom in Berlin in einem 19" Schrank. Zugriff erfolgt über SSH, vom Arbeitsplatz der Administratoren. Zusätzlich ist ein Monitor und eine Tastatur mit Wechselschalter an die Server angeschlossen, um live Logfiles auswerten zu können. Internetzugriff erfolgt über die zweite Netzwerkkarte des Installationservers, diese bekommt eine lokale IP-Adresse, und kann darüber auf den netzinternen Proxy-Server in Frankfurt zugreifen.



4.2 Inbetriebnahme des Installationservers

Der Installationsserver wird von den aktuellsten Quellen aus dem Internet mit Hilfe von Bootdisketten installiert. Dabei ist die unter 2.2 beschriebene Methode zum Einsatz gekommen, Quelle ist der RedHat-FTP-Server (Internet). Eine zuvor erstellte *ks.cfg* mit Minimalkonfiguration der Pakete ist benutzt worden. Partitioniert habe ich den Server in vier Partitionen: */swap*, */*, */boot* und */data*. Nach dem Reboot des Servers mussten noch Einträge in die */etc/resolv.conf* vorgenommen werden, ein Benutzer malte angelegt und dann die Installationspakete vom RedHat-FTP-Server mit *nftp* auf die */data* Partition gespiegelt werden.

4.3 Installation DHCP-Server

Mit *rpm -ivh /data/RedHat/dhcp-3.0* wird der DHCP-Server installiert. Die Konfigurationsdatei */etc/dhcpd.conf* muss auf die Gegebenheiten angepasst werden, die fertige *dhcpd.conf* finden sie im Anhang.

4.4 Installation FTP-Server

`rpm -ivh /data/pub/RedHat/vsftpd` installiert den von RedHat empfohlenen FTP-Server. Durch `chroot_local_users=yes` in der Konfigurationsdatei `/etc/vsftpd/vsftpd.conf` erreicht man die notwendige Sicherheit um einen FTP-Server im Netz zu betreiben. Der anonyme FTP-Account zeigt default nach `/var/ftp/`. Dort wurde ein neues Verzeichnis angelegt mit `mkdir /var/ftp/pub/` und anschliessend die `/data`-Partition mit `mount -bind /data /var/ftp/pub` gemountet. Ein Eintrag in der `/etc/fstab` sichert den Mountvorgang auch nach einem Reboot. Weitere Informationen entnehmen Sie bitte der `fstab` und der `/etc/vsftpd/vsftpd.conf` im Anhang, sowie der Projektseite <http://vsftpd.beasts.org/>.

4.5 NFS-Server Freigabe

Als Nächstes wird die `ks.cfg` und der Installationskernel über ein Netzlaufwerk, das vom Client gemountet wird, freigegeben. Beim Mountrequest eines Clients wird der Inhalt der `/etc/exports` gechecked und mit der IP-Adresse / Namen des anfragenden Clients verglichen. Dazu benötigt die `/etc/exports` einen Eintrag, der so gewählt ist, dass der Client im Rechenzentrum diese Freigabe benutzen kann. Zur Installation wird eine bestimmte IP-Adresse genutzt, diese ist in der `/etc/exports` einzutragen. Weiterhin schauen Sie sich bitte die `/etc/exports` im Anhang an.

4.6 Installation TFTP-Server

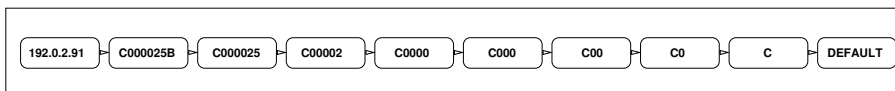
PXE benötigt einen TFTP-Server, der das Trivial File Transfer Protokoll unterstützt. Benutzer können nicht autorisiert werden. Hier wird nur der Kernel und die Ramdisk für den Client hinterlegt. Mit `rpm -ivh /data/RedHat/tftp-server-0.32-4` wird der TFTP-Server installiert. In der Konfigurationsdatei `/etc/xinetd.d/tftp` wird `disable=no` gesetzt, so das ab sofort vom `xinetd` Anfragen an den TFTP-Server weitergegeben werden. Des weiteren wird noch der Eintrag für die TFTP-Freigabe benötigt: `server args = -s /tftpboot`. Siehe Anhang.

4.7 Bootloader bereitlegen

Um per PXE den Kernel zu laden, muss ein Bootloader benutzt werden. PXE kann Bootloader bis zu 64 KB laden. Das Syslinux-Projekt von H. Peter Advin bietet eine gute Grundlage. Also wird der Bootloader geladen, der dann wiederum den Kernel laden kann. Das Vorhandensein einer Konfigurationsdatei für den Bootloader entscheidet über die Art und Weise des Bootens, ob der Client installiert wird, oder lokal von seiner Festplatte bootet.

Im Verzeichnis `/tftpboot/` ist der Bootloader `pxelinux.0` plaziert. Im Verzeichnis `/tftpboot/pxelinux.cfg/` sucht der PXE-linux Bootloader nach seiner Konfigurationsdatei. Damit mehrere Clients gleichzeitig installiert werden können, wird die Konfigurationsdatei einer IP-Adresse zugeordnet. PXElinux sucht auf dem Server nach der Client-Konfigurationsdatei in folgender Weise:

1. Zuerst sucht er nach seiner eigenen IP-Adresse in "upper case hexadecimal", z.B. 192.168.1.1 => C0A80101
2. Findet er keine Datei mit diesem Namen, entfernt er eine Hex-Ziffer und versucht es nochmal, als Beispiel :



Die default-Konfigurationsdatei ist so vorbereitet, dass sie automatisch das lokale Betriebssystem des Clients bootet. Mit dem C-Programm `gethostip` von H. Peter Advin kann eine IP-Adresse in Hex-Ziffern umgewandelt werden. Das Programm finden Sie im Anhang, weitere Informationen auf der Syslinux-Projektseite <http://syslinux.zytor.com/>.

4.8 Installationsquellen aktualisieren

Für jede Installation sind die Installationsquellen das A und O. Um möglichst aktuelle Quellen zu benutzen, wurde der komplette Pfad des RedHat-FTP-Servers lokal gespiegelt (siehe 3.1). Um eine kontinuierliche Aktualität der einzelnen Pakete zu gewährleisten, kommt das Programm `Mirror` zum Einsatz, um die Quellen täglich mithilfe eines `Cronjobs` zu aktualisieren. Dabei werden zwei gleiche Verzeichnisse angelegt, und mittels eines Skriptes verglichen. Wenn neue Pakete von `Mirror` geladen werden, erkennt das Skript die aktuelle Version, und verschiebt das outdated-Paket nach `/data/Redhat9/old_i386/`. Dann wird das aktuelle Paket in den `/data/Redhat9/Redhat/RPMS/`-Ordner verschoben. Zusätzliche Sicherheit bietet RedHats RPM-GPG-KEY, der in den Header der Paket-Binaries von RedHat eingefügt wird. `Mirror` vergleicht die zu besorgenden Pakete mit dem lokal auf dem System eingetragenen GPG-KEY (`rpm -import /usr/share/rhn/RPM-GPG-KEY`), um auszuschliessen, das die Pakete manipuliert wurden. Für die Installation benötigt

Anaconda (das RedHat-Installationsprogramm) die *hdlist*-Datei, sie enthält Informationen über die Pakete (Versionen etc.) des aktuellen Installationsverzeichnis. Zur Generierung dieser Datei benötigt der Server das Paket *anaconda-runtime* (*rpm -ivH /data/RedHat9/RedHat/RPMS/anaconda-runtime*).

Somit kann gewährleistet werden, dass Kunden "gute" Pakete bekommen und ihre alten Versionen der Original-Pakete noch auf dem FTP-Server vorfinden, wenn schon aktualisierte Pakete der Distribution herausgegeben wurden. Die Konfigurationsdatei für das Programm *Mirror* und das *update.sh*-Skript finden Sie im Anhang.

5 Logging

5.1 ...der Installationen

Die Installationen werden vom FTP-Server mitgeloggt. Anhand der */var/log/vsftpd.log*-Dateien kann der Administrator direkt nachvollziehen, welche Pakete gerade installiert werden. Durch Logrotate werden die Logdateien im wöchentlichen Turnus weiterverschoben, komprimiert und vier Wochen aufbewahrt. Die *logrotate.conf* finden Sie im Anhang. In der Datei */root/install.log* auf dem Client kann der Administrator zusätzlich eine Auflistung der Pakete inklusive Versionen zum Installationszeitpunkt nachlesen, siehe Anhang.

5.2 ...des Paketupdates auf dem Server

Zu diesem Zwecke legt das Programm *Mirror* eine Logdatei im */data/updates/9/i386/*-Verzeichnis an. Die Datei heisst *.mirror*, durch die Positionierung im Updatesverzeichnis gelingt es, dem Anwender ein Changelog-ähnliches Dokument zu präsentieren. Ein Beispiel finden Sie im Anhang.

5.3 ...der FTP-Serverzugriffe

Wie unter Punkt 5.1 beschrieben, werden Client-Zugriffe auf den FTP-Server für vier Wochen mitgeloggt. Bei Bedarf kann das beliebig erweitert werden. Somit kann im Worst-Case nachvollzogen werden, welche Pakete Probleme verursachen. Zu statistischen Zwecken kann die FTP-Logdatei ebenfalls ausgewertet werden.

6 Härten des Systems

6.1 Generelles

Es gilt abzuwägen, welche Funktionalitäten der Server bereitstellen soll, und wie sicher er sein möge. Je mehr Dienste man ansprechen kann, desto unsicherer wird das ganze System. Je sicherer man den Server macht, umso umständlicher wird dessen Bedienung, umso mehr Passwörter benötigt man, um an relevante Informationen zu gelangen, oder um vom Server angebotene Dienste zu benutzen. Implementierung von Alarmanlagen würden den Projektrahmen sprengen. Ebenfalls schützen solche hochsicheren Systeme nur vor Angriffen aus dem Internet, gegen interne Attacken sind auch sie meistens nutzlos. Jede Barriere (z.B. die Implementierung eines Key-Servers) kann umgangen werden. DoS-Attacken können auf jedes System gefahren werden, dessen Dienste Anfragen entgegen nehmen.

Durch die Platzierung des Servers innerhalb des Admin-LANs des Rechenzentrums werden Angriffe nahezu ausgeschlossen.

Um noch weitere Sicherheit zu bekommen, ist es möglich, die Dienste auf mehrere Rechner zu verteilen. Wird der DHCP-Server gehackt, können Kunden weiterhin auf den FTP-Server zugreifen. Redundante Systeme sind in der Realisierung nicht vorgesehen, ist aber ohne Weiteres möglich, es ändern sich lediglich einige Konfigurationsdateien.

Weiterführende Literatur gibts unter:

<http://www.tldp.org/LDP/solrhe/Securing-Optimizing-Linux-The-Ultimate-Solution-v2.0.pdf>

6.2 mehrere Netzwerkkarten-mehrere Netze

Durch die zwei Netzwerkkarten besteht die Möglichkeit, die Installations-Dienste des Servers nur im Admin-LAN zur Verfügung zu stellen. Mindestens der für Kunden zugängliche FTP-Server muss aber in der DMZ stehen, ebenfalls benötigt der Server Internetzugang, um das Update der Installationsquellen durchzuführen.

6.3 Firewall

Durch die Installation des *IP-Tables*-Frontend *Shorewall* lässt sich mit einfachen Konfigurationsdateien ein effektiver kernelbasierter Paketfilter implementieren. Die Einstellung erfordert ein sorgfältig geplantes Vorgehen. Eine gute Anleitung dazu gibt es unter www.shorewall.net, die bei jeder Implementierung wieder befolgt werden sollte.

Da die Firewall und deren Einrichtung hier den Rahmen des Projektes sprengen würde, wird nur kurz auf wichtige Konfigurationsdateien hingewiesen.

- */etc/shorewall/rules* - hier werden Ports freigegeben, geforwarded usw.
- */etc/shorewall/zones* - legt die Zonen fest, für die Unterscheidung der Netze (LAN/WAN/DMZ)
- */etc/shorewall/policy* - grundsätzliche Filterregeln
- */etc/shorewall/interfaces* - Netzwerkkarten und die an ihnen angeschlossenen Netze
- */etc/shorewall/routestopped* - falls die Firewall gestoppt wird, können die hier eingetragenen Hosts weiterhin auf den Rechner zugreifen (z.B. per *SSH*).

6.4 Benutzer

Es ist notwendig einen Wartungsaccount anzulegen, zur Sicherheit ohne Passwort (kein Einloggen möglich) und mit einer *restricted shell*. Mit diesem Account läuft zum Beispiel das *update.sh*-Skript.

RedHat hat nach der Standardinstallation noch Benutzer, die entfernt werden müssen, um keinen unauthorisierten Zugriff auf das System zu ermöglichen. Hierzu zählen: *lp, news, sync, uucp, games, gopher, rpc, nscd, und pcap*.

6.5 überflüssige Pakete entfernen

Funktionen, die der Server nicht für den Betrieb benötigt, werden am besten gleich deinstalliert.

RedHats Minimalkonfiguration installiert Pakete, die nicht benötigt werden, wie etwa *sendmail*.

7 aufgetretene Probleme

7.1 Bootloader

Während des Projektes traten diverse Probleme auf, auf die in der technischen Dokumentation (Punkte 3-6) hingewiesen wird. Ein Problem wird hier nochmals gesondert aufgeführt: die Konfiguration des Bootloaders ist knifflig. Wird der Bootloader nicht nach der Installation entfernt, installiert sich der Server nach dem Reboot der erfolgreichen Installation in einer Endlosschleife, da immer wieder (ist als *FirstBootDevice* eingetragen) über PXE nach einem DHCP-Server gefahndet wird. Da niemand neben der Installation stehen soll, sowieso kein Bildschirm an dem zu installierenden Server angeklemt ist, musste eine Lösung geschaffen werden, die dieses Problem umgeht. Eine ausgiebige Recherche im Internet ergab eine Lösung, die das Problem behebt: In die *ks.cfg* wird ein Paket eingetragen, das der zu installierende Server anfordert, das aber auf dem Installationsserver nicht existiert. Auf dem Installationsserver beobachtet ein Daemon das Logfile des TFTP-Servers (*/var/log/messages*), bis er die Fehlermeldung des nicht vorhandenen Paketes entdeckt. In diesem Moment entfernt der Daemon die Bootloader-Konfigurationsdatei für den Client, und bei dessen nächster Anfrage nach der Installation findet er jetzt keinen Bootloader mehr, der ihm den Installationskernel anzeigt. Stattdessen greift er auf den Standard-Bootloader zurück, der dem Installationsclient befiehlt, von der lokalen Festplatte zu booten.

7.2 bei der *ks.cfg*

Partitionierungsdaten müssen eingetragen sein, da sonst die Installation eine Eingabe des Benutzers erwartet. Die Netzwerkkarte, über die installiert werden soll, muss eingetragen werden. Das Paket *TFTP* muss mitinstalliert werden, um dem Client zu ermöglichen, nach Abschluss der Installation (via *TFTP*) die Datei *kickstart_end* anzufordern (Eintrag in *post-install*-Routine).

7.3 Serverhardware

Ein schwieriges Problem ist die Serverhardware, deren Netzwerkkarten im Besonderen. Mal ist die onboard-NIC PXE-fähig, mal die Steckkarte. Manche sind gar nicht PXE-fähig. BootPROMs können nicht einfach nachgerüstet werden, da Garantieverlust des Herstellers droht. In einigen Servern ist die erste Steckkarte eth0 (erste NIC bei Linux, muss in der *ks.cfg* angegeben werden), bei anderen wiederum die onboard-NIC. Nach einiger Tüftelei mit PXE-Bootloadern auf Disketten bei nicht PXE-fähigen Servern fiel die Entscheidung, diese Server weiterhin mit BootCD zu installieren. Beim Booten muss dann auf der Konsole eine Option an den Kernel übergeben werden: *ks=10.162.xxx.55*, ebenfalls muss manuell die IP-Adresse vergeben werden. Danach folgt die oben beschriebene Routine, der Server installiert ebenfalls mit der vorbereiteten *ks.cfg*. Nach beendeter Installation bootet der Server von CD, die Installationsroutine bleibt aber gleich bei der ersten Eingabeaufforderung stehen. Der Server wird manuell resettet, die CD entnommen und das fertige System gestartet.

8 Rollout

Nach ausgiebiger Testphase wurde der Installations-Server im Rechenzentrum mit Herrn Wiebe implementiert, der für die Infrastruktur der Server zuständig ist. Die zwei Netzwerkinterfaces wurden in die Netzwerkstruktur integriert, indem mit Herrn Werner (Team Network) ein neues V-LAN im Administrations-LAN erstellt wurde, exklusiv für die Installation der Unix-Server. Zu diesem einfachen aber effektiven Sicherheitssystem riet Herr Werner, nachdem klar wurde, wie einfach jemand eine falsche MAC-Adresse im Bootloader für einen zu installierenden Server eingeben kann. Dadurch kann unter Umständen grosser Schaden an Kundenrechnern entstehen, wenn dieser im falschen Moment neu bootet. Durch die Einrichtung des V-LANs muss nun der neu zu installierende Server mit dem Administrations-LAN verbunden werden, und von dem Netzwerkteam auf dem zuständigen Switch in das Unix-Installations-VLAN gehoben werden. Erst dann ist eine Installation möglich. Von dem Zeitpunkt an steht der Server für Installationen bei der Colt Telecom GmbH im Rechenzentrum zur Verfügung. Da der Rollout erst zwei Wochen vor Ende meines Praktikums stattfand, habe ich mit dem Installations-Server nur einige wenige Kundenmaschinen installieren können.

8.1 Mitarbeitereinweisung

Die Mitarbeitereinweisung fand bei der Colt Telecom GmbH im Konferenzraum statt. Da der Kunde gleichzeitig auch der Betreuer des Projektes ist, hielt sich der Umfang im kleinen Rahmen. Technische Besonderheiten, die aufgrund der Freiheit entanden sind, die bei der Implementation gewährt wurde, sind geklärt worden (Pfadangaben, Mirror-update, Webinterface). Den Administratoren ist eine technische Dokumentation übergeben worden, die auch Bestandteil dieser Projektdokumentation geworden ist.

9 Qualitätssicherung

Zur Qualitätssicherung wurde das gesammelte Know-How auf einer CD zusammengestellt, und mit der technischen Dokumentation an die Administratoren ausgehändigt. Wichtige Skripte, Dokumentationen über Bootloader-Konfiguration und nicht in der Standard-Distribution enthaltene Software befand sich auf der CD. Weiterhin ist in dem Server ein Hardware-RAID 1 zwecks Datensicherung enthalten, das bei Ausfall einer Festplatte das System leicht wiederhergestellt werden kann. Der Server ist in der unternehmensweiten Überwachungssoftware integriert worden, so dass bei Ausfall sofort der Wachschatz des Rechenzentrums benachrichtigt wird. Auf ein Backup auf Band habe ich aufgrund der getroffenen Sicherungsmassnahmen verzichtet, ist das Know-How erst einmal vorhanden, ist die Implementierung eines Installationssservers nicht weiter aufwändig und passiert an einem Arbeitstag.

10 Soll-Ist-Vergleich

Die geforderten Rahmenbedingungen konnten in der vorgegebenen Zeit nicht ganz erfüllt werden. Daher ist die Implementierung der Firewall aus meinem Projekt gestrichen worden und hier nur kurz angeführt. Vorher war erkennbar, das das den Projektrahmen von 35 Stunden sprengen würde. Sonst bin ich zufrieden mit der von mir geleisteten Arbeit, der beste Beweis ist die Integration des Installationssservers im Rechenzentrum und die Nutzung durch die Administratoren. Bei der Planung des Projektes hätte mehr Zeit auf ein Pflichtenheft verwendet werden sollen, das hätte viele Fragen beantwortet, die im Nachhinein nur zeitintensiv zu klären waren. Zeitersparend war, die Dokumentation mit \LaTeX anzufertigen.

11 Erklärung

Hiermit erkläre ich, die obige Dokumentation selbst erstellt und das Projekt bei der Colt Telecom GmbH durchgeführt zu haben.

Berlin, 1. April 2004

Malte Eismann

A Anhang

A.1 Glossar

Begriff	Erklärung
BootPROM	Flashspeicher, ProgrammableReadOnlyMemory, enthält Software für PXE
<i>chroot</i>	die change root stellt ein virtuelles System dar, in dem Prozesse eingesperrt werden können
<i>cronjob</i>	der Cronjob ist ein Daemon, um Aufgaben regelmässig zeitgesteuert auszuführen
desaster recovery	engl.: Notfall-Wiederherstellung
etherboot	Open-Source Projekt, ermöglicht PXE-Software von Diskette in Speicher zu laden
<i>FirstBootDevice</i>	im Bios eingetragener Device, von dem ein System zuerst bootet
Frontend	Schnittstelle zwischen Benutzer und einem Programm
HE	Höheneinheit für IT-Geräte, eine HE=4.425 cm
<i>ip-tables</i>	kernelbasierter Paketfilter
Kickstart	unbeaufsichtigte Installation, vergleichbar mit Microsofts' Unattended Installation
<i>ks.cfg</i>	Konfigurationsdatei für die unbeaufsichtigte Installation
NFS	Network File System (Unix), Freigaben können als Laufwerke gemountet werden
NIC	Network-Interface-Card, engl. für Netzwerkkarte
onboard	auf dem Mainboard integriert
<i>Post-Install-Skript</i>	in der <i>ks.cfg</i> enthaltenes Skript, das nach der Installation ausgeführt wird
<i>Pre-Install-Skript</i>	Skript in der <i>ks.cfg</i> , das vor der Installation abgearbeitet wird
PXE	Pre-eXecution-Environment, dient zum Booten aus dem Netzwerk
<i>restricted shell</i>	eingeschränkte Form der <i>Bash-Shell</i>
<i>SSH</i>	SecureShell, ähnlich <i>telnet</i> , aber verschlüsselt
TFTP	Trivial File Transfer Protokoll
V-LAN	Virtuelles-LAN, in dem Rechner zu einem virtuellen Netzwerk zusammengefasst werden
Xinetd	Daemon, der Ports überwacht und bei Anfragen auf einen Port die zugehörige Software startet

A.2 Literaturquellen

Name	Quelle
RedHat	www.redhat.com , ftp://ftp.redhat.com
Developer Manual Etherboot	http://etherboot.sourceforge.net
VS-FTP-Serverprojekt	http://vsftpd.beasts.org
Syslinux-Projekt	http://syslinux.zytor.com
Linux Grundlagen und Security	www.tldp.org
Shorewall Firewall	www.shorewall.net

A.3 Konfigurationsdateien

Einige firmeninterne Angaben wie IP-Adressen und/oder Benutzernamen wurden unkenntlich gemacht.

A.3.1 ks.cfg

```
# Kickstart Revision: $Revision: 2.12 n#####
# Standort: 1 # 0
#
lang en_US
langsupport en_US
keyboard de-latin1-nodeadkeys
mouse none
timezone Europe/Berlin
zerombr yes
rootpw xxx
reboot
text
#bootloader --location=mbr --useLilo
bootloader --location=mbr
install
url --url http://10.172.xxx.2/rh9
clearpart --all --initlabel
#####
# Hier befinden sich die Partitionierungsdaten
#####
part /boot --fstype ext3 --size 50
part swap --size 1000
# if 36g... /var should be 3000
part /var --fstype ext3 --size 1500
part /usr --fstype ext3 --size 1500
part /home --fstype ext3 --size 1000
part /tmp --fstype ext3 --size 500
part / --fstype ext3 --size 500
part /data --fstype ext3 --size 1 --grow
auth --useshadow --enablemd5
skipx
network --device eth1 --bootproto static --ip INSTALL_IP --netmask INSTALL_NETMASK --gateway INSTALL_GATEWAY
--nameserver INSTALL_NAMESERVER
#####
# zu installierende Pakete
#####
%packages
hwdata
man-pages
rmt
dump
basesystem
chkconfig
--snip-
#hier fehlt die vollständige Liste der Pakete, minimal 172 Stück
--snap-
openssl-devel
up2date
#####
# Start der Postinstallationroutine
#####
#####
# Konfiguration der /etc/rc-Dateien
```

```
#####
%post --interpreter /usr/bin/perl
'chkconfig --del autofs';
'chkconfig --del lpd';
'chkconfig --del kudzu';
'chkconfig --del isdn';
'chkconfig --del apmd';
'chkconfig --del gpm';
'chkconfig --del netfs';
'chkconfig --del apmd';
'chkconfig --del xfs';
'chkconfig --del portmap';
'chkconfig --del nfslock';
'chkconfig --del ipchains';
'chkconfig --del rhnsd';
'chkconfig --level 3 httpd on';
'chkconfig --level 3 iptables off';
'chkconfig --level 3 portmap off';
if (0)
{
'chkconfig --level 3 mysqld on';
}
#####
# Einige Sourcen werden noch benoetigt:
#####
'mkdir /usr/local/src';
'wget http://10.172.xxx.2/extras/bmc_linux.tgz -O /usr/local/src/bmc_linux.tgz >> /root/wget.log 2>&1';
'wget http://10.172.xxx.2/extras/report.sh -O /usr/local/bin/report.sh >> /root/wget.log 2>&1';
'wget http://10.172.xxx.2/extras/get_date -O /etc/cron.daily/get_date >> /root/wget.log 2>&1';
'wget http://10.172.xxx.2/extras/icpcon -O /sbin/icpcon >> /root/wget.log 2>&1';
'chmod u+x /etc/cron.daily/get_date';
'chmod u+x /sbin/icpcon';
#####
# Installation der Backupsoftware je nach Installations-
# quelle.
#####
# die alte 3.4er Version von Oliver
'rpm -ivh http://10.172.xxx.2/rpms/VERITAS-NB-3.4-6.i386.rpm';
'chmod o-w-r-x -R /usr/opensv/';
#####
# Konfiguration der Backupsoftware
#####
'mkdir -p /usr/opensv/netbackup';
open(F,">/usr/opensv/netbackup/bp.conf");
print F "SERVER = BACKUP_SRV";
print F "CLIENT = FQDN_MAN";
close(F);
open(F,"> /etc/hosts");
print F "INSTALL_IP FQDN_MAN SERVERNAME";
print F "PUBLIC_IP FQDN_PUB";
print F "TIMESRV_IP TIMESRV_NAME";
close(F);
'useradd kenobi -g wheel';
'echo xxx | passwd --stdin kenobi';
'useradd monitor';
'echo xxx | passwd --stdin monitor';
#####
# Konfiguration von Apache
```

```

#####
#####
# Konfiguration von MySQL
#####
#####
# Konfiguration von xinetd
# 1. Backupdaemon
#####
open(F,">/etc/xinetd.d/bpcd");
print F <<EOB;
service bpcd
{
socket_type = stream
protocol = tcp
wait = no
user = root
server = /usr/opensv/netbackup/bin/bpcd
disable = no
only_from = 10.172.1.1
}
EOB
close(F);
#####
# Erstellung von festen Routingeintraegen
#####
open(F,">/etc/sysconfig/network-scripts/route-eth1");
print F "10.0.0.0/8 via INSTALL_GATEWAY"
print F "TIMESRV_IP/32 via INSTALL_GATEWAY"
close(F);
'mv /etc/sysconfig/network /etc/sysconfig/network.old';
open(F,">/etc/sysconfig/network");
print F "NETWORKING=yes";
print F "HOSTNAME=SERVERNAME";
print F "GATEWAY=DEFAULT_GATEWAY";
print F "GATEWAYDEV="eth0";
close(F);
#####
# Konfiguration des Timeservers
#####
open(F,">/etc/ntp/step-tickers");
print F "TIMESRV_IP";
close(F);
'cp /etc/ntp.conf /etc/ntp.bak';
'sed "s#^server.*127.127.1.0.*#\&\ =server.*TIMESRV_IP\ prefer#" /etc/ntp.conf > /etc/ntp.temp';
'tr '=' '\n' </etc/ntp.temp >/etc/ntp.conf';
'rm /etc/ntp.temp';
#####
# Konfiguration des Nameservice
#####
open(F,">/etc/resolv.conf");
print F "domain hosting.blm.de.colit-isc.net";
print F "nameserver INSTALL_NAMESERVER";
#print F "nameserver 62.96.xxx.228";
print F "search hosting.blm.de.colit-isc.net private.blm.de.colit-isc.net";
close(F);
#####
# /etc/ld.so.conf
#####

```

```

open (F,">/etc/ld.so.conf");
print F "/usr/kerberos/lib";
print F "/usr/lib";
close(F);
#####
# Konfiguration der Netzwerkkarten
#####
open(F,">/etc/sysconfig/network-scripts/ifcfg-eth0");
print F <<ETH0;
DEVICE=eth0
BOOTPROTO=static
IPADDR=PUBLIC_IP
GATEWAY=DEFAULT_GATEWAY
NETMASK=PUBLIC_NETMASK
ONBOOT=yes
ETH0
close(F);
open(F,">/etc/sysconfig/network-scripts/ifcfg-eth1");
print F <<ETH1;
DEVICE=eth1
BOOTPROTO=static
IPADDR=INSTALL_IP
NETMASK=INSTALL_NETMASK
ONBOOT=yes
ETH1
close(F);
#####
# haerten des systems
#####
## die besonderen packete wollen auch alle installiert werden
open(F, "> /root/pre-install.sh");
print F <<EOPRE;
#!/bin/sh
# echo "[ \$LOGNAME == "kenobi" ] ;echo " [ be aware of the immutability ] " >> /etc/profile
#mount -o remount,rw /usr
cd /usr/local/src
echo "installing hurl.."
rpm -ivh http://10.172.xxx.2/extras/colt_hurl.rpm
/sbin/hurl-colt.py -i
sed s/read\ nothing//g < /var/local/updates/go.sh > /var/local/updates/go_install.sh
chmod 755 /var/local/updates/go_install.sh
/var/local/updates/go_install.sh
rm /var/local/updates/go_install.sh
rm -f /var/local/updates/*
cd /usr/local/src/
tar xzfv bmc_linux.tgz
cd Linux
./install_linux.patrolagent.pl
echo cKYZ8fFw | passwd --stdin smop
passwd -d patrol
userdel lp
userdel news
userdel sync
userdel uucp
userdel games
userdel gopher
userdel rpc
userdel nscd

```

```

userdel pcap
sed s/\\sbin\\nologin/\\bin/false/g < /etc/passwd > /etc/passwd.new
mv /etc/passwd.new /etc/passwd
cd /usr/local/src/
if [ -e proftpd.tar.bz2 ] ; then
echo "installing proftpd..."
tar xjfv proftpd.tar.bz2
cd proftpd*
./INSTALL.sh
fi
cd /usr/local/src/
if [ -e rm_rpm.tar.gz ] ; then
echo "removing useless rpms..."
tar xzfv rm_rpm.tar.gz
cd rm_rpm
./remove_rpm.sh
fi
cd /usr/local/src/
if [ -e patches.tar.gz ] ; then
echo "patching..."
tar xzfv patches.tar.gz
cd patches
bash < ./install
fi
cd /usr/local/src
if [ -e colt-webserver.tar.gz ] ; then
echo "installing Colt-Virtual-Webserver"
tar xzfv colt-webserver.tar.gz
cd colt-webserver
./install_lamp.sh
apachectl restart
fi
# cleaning up
#m -rf /var/www/html/manual
#rm -f /etc/grub.conf
#m -f /etc/inetd.conf
echo "" > /etc/issue
echo "" > /etc/issue.net
chmod 700 /etc/init.d/*
#lilo -c
mkdir /data/home
useradd kick
echo kick | passwd -stdin kick
#usr/bin/chatr +i /etc/passwd
#usr/bin/chatr +i /etc/shadow
/usr/bin/chatr +i /etc/services
/usr/bin/chatr +i /etc/xinet.d/*
#/usr/bin/chatr +i /etc/lilo.conf
/bin/chmod a-s /usr/bin/chage
/bin/chmod a-s /usr/bin/gpasswd
/bin/chmod a-s /usr/bin/wall
/bin/chmod a-s /usr/bin/chfn
/bin/chmod a-s /usr/bin/chsh
/bin/chmod a-s /usr/bin/newgrp
/bin/chmod a-s /usr/bin/write
/bin/chmod a-s /usr/sbin/usernetctl
/bin/chmod a-s /bin/mount
/bin/chmod a-s /bin/umount

```

```

/bin/chmod a-s /bin/ping
/bin/chmod a-s /sbin/netreport
/bin/bash /usr/local/bin/report.sh
rm /usr/local/bin/report.sh
EOPRE
close(F);
#####
# Konfiguration "Message of the Day"
#####
open(F, ">/etc/motd");
print F <<EOM;
*****
NOTICE TO USERS
This computer system is for authorized use only. Users (authorized or
unauthorized) have no explicit or implicit expectation of privacy.
Any or all uses of this system and all files on this system may be
intercepted, monitored, recorded, copied, audited, inspected, and disclosed
to authorized site and law enforcement personnel.
By using this system, the user consents to such interception, monitoring,
recording, copying, auditing, inspection, and disclosure at the discretion
of authorized site.
Unauthorized or improper use of this system may result in administrative
disciplinary action and civil and criminal penalties. By continuing to use
this system you indicate your awareness of and consent to these terms and
conditions of use. LOG OFF IMMEDIATELY if you do not agree to the conditions
stated in this warning.
*****
EOM
close F;
#####
# firstboot initskript
open (FIRSTBOOT, "> /etc/rc.d/init.d/firstboot");
print FIRSTBOOT <<EOFB;
#!/bin/sh
#
# Init File for firstboot
#
# chkconfig: 2345 55 25
# description: Firstboot
#
# processname: noone
#
# source function library
. /etc/rc.d/init.d/functions
RETVAL=0
# see how we were called
case "$1" in 'start')
/bin/bash < /root/pre-install.sh
mv /etc/rc.d/init.d/firstboot /root/firstboot.done
rm /root/pre-install.sh
rm /root/wget.log
reboot
;;
*)
echo "usage: { start }"
exit 1
;;
esac

```

```

exit 0
EOFB
close (FIRSTBOOT);
'ln -s /etc/rc.d/init.d/firstboot /etc/rc.d/rc3.d/S99firstboot';
'chmod u+x /etc/rc.d/init.d/firstboot';
# testweise um zu sehen, ob das hier auch schon tut
#'lilo -c';
#####
# default firewall
open (FIREWALL, ">/etc/sysconfig/iptables");
print FIREWALL <<EOFW;
# Generated by iptables-save v1.2.5 on Wed Mar 26 12:57:11 2003
# COLT STD FILTER
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:DNS_IN - [0:0]
:DNS_OUT - [0:0]
:LOGDROP - [0:0]
:MYSQL_IN - [0:0]
:PINGPONG - [0:0]
-A INPUT -j DNS_IN
-A INPUT -j PINGPONG
-A INPUT -j MYSQL_IN
-A OUTPUT -j DNS_OUT
-A DNS_IN -s 10.162.xxx.1 -p udp -m udp --sport 53 --dport 1024:65535 -j ACCEPT
-A DNS_IN -s 10.162.xxx.1 -p tcp -m tcp --sport 53 --dport 1024:65535 ! --tcp-flags SYN,RST,ACK SYN -j ACCEPT
-A DNS_IN -p tcp -m tcp --sport 53 --dport 1024:65535 -j LOGDROP
-A DNS_IN -s 62.96.xxx.228 -p udp -m udp --sport 53 --dport 1024:65535 -j ACCEPT
-A DNS_IN -s 62.96.xxx.228 -p tcp -m tcp --sport 53 --dport 1024:65535 ! --tcp-flags SYN,RST,ACK SYN -j ACCEPT
-A DNS_IN -p tcp -m tcp --sport 53 --dport 1024:65535 -j LOGDROP
-A DNS_IN -s 62.96.xxx.235 -p udp -m udp --sport 53 --dport 1024:65535 -j ACCEPT
-A DNS_IN -s 62.96.xxx.235 -p tcp -m tcp --sport 53 --dport 1024:65535 ! --tcp-flags SYN,RST,ACK SYN -j ACCEPT
-A DNS_IN -p tcp -m tcp --sport 53 --dport 1024:65535 -j LOGDROP
-A DNS_OUT -d 10.162.xxx.1 -p udp -m udp --dport 53 -j ACCEPT
-A DNS_OUT -d 10.162.xxx.1 -p tcp -m tcp --dport 53 -j ACCEPT
-A DNS_OUT -d 62.96.xxx.228 -p udp -m udp --dport 53 -j ACCEPT
-A DNS_OUT -d 62.96.xxx.228 -p tcp -m tcp --dport 53 -j ACCEPT
-A DNS_OUT -d 62.96.xxx.235 -p udp -m udp --dport 53 -j ACCEPT
-A DNS_OUT -d 62.96.xxx.235 -p tcp -m tcp --dport 53 -j ACCEPT
-A LOGDROP -m limit --limit 4/sec -j LOG --log-prefix "[IPTABLES] packet died: "
-A LOGDROP -j DROP
-A MYSQL_IN -d 213.61.xxx.24 -p tcp -m tcp --sport 1024:65535 --dport 3306 -j LOGDROP
-A PINGPONG -p icmp -m icmp --icmp-type 17 -j LOGDROP
-A PINGPONG -p icmp -m icmp --icmp-type 18 -j LOGDROP
-A PINGPONG -p icmp -m icmp --icmp-type 13 -j LOGDROP
-A PINGPONG -p icmp -m icmp --icmp-type 14 -j LOGDROP
COMMIT
EOFW
close(FIREWALL);

```

A.3.2 dhcpd.conf

```

ddns-update-style none;
subnet 10.162.xxx.0
netmask 255.255.255.0
{

```

```

option routers 10.162.xxx.254;
option subnet-mask 255.255.255.0;
option domain-name "workgroup";
range 10.162.xxx.56 10.162.xxx.56;
default-lease-time 21600;
max-lease-time 43200;
allow unknown-clients;
}
group
{
default-lease-time 7200;
max-lease-time 64800;
host CORFU
{
hardware ethernet 00:XX:6E:1C:BC:D9;
fixed-address 10.162.xxx.56;
next-server 10.162.xxx.55;
# filename "pxelinux.0";
filename "0AC0640C";
option root-path "initrd-taroon.img";
}
}

```

A.3.3 vsftpd.conf

```

# This is vsftpd.conf
# maintained by Malte Eismann <malte@fuldastrasse.de>
# implemented @ Colt Telekom, Berlin
anonymous_enable=YES
local_enable=YES
write_enable=YES
local_umask=022
#anon_upload_enable=YES
#anon_mkdir_write_enable=YES
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
#chown_uploads=YES
#chown_username=whoever
#xferlog_file=/var/log/vsftpd.log
xferlog_std_format=YES
#idle_session_timeout=600
#data_connection_timeout=120
#nopriv_user=ftpsecure
#async_abor_enable=YES
#ascii_upload_enable=YES
#ascii_download_enable=YES
#ftpd_banner=Welcome to blah FTP service.
#deny_email_enable=YES
#banned_email_file=/etc/vsftpd.banned_emails
#chroot_list_enable=YES
#chroot_list_file=/etc/vsftpd.chroot_list
ls_recurse_enable=YES
pam_service_name=xxxxxx
userlist_enable=YES
#enable for standalone mode listen=YES
tcp_wrappers=NO chroot_local_user=YES

```

A.3.4 fstab

```
LABEL=/ / ext3 defaults 1 1
LABEL=/boot /boot ext3 defaults 1 2
LABEL=/data /data ext3 defaults 1 2
none /dev/pts devpts gid=5,mode=620 0 0
none /proc proc defaults 0 0
none /dev/shm tmpfs defaults 0 0
/dev/sda3 swap swap defaults 0 0
/dev/cdrom /mnt/cdrom udf,iso9660 noauto,owner,kudzu,ro 0 0
/dev/fd0 /mnt/floppy auto noauto,owner,kudzu 0 0
/data/ /var/ftp/pub/ ext3 ro,bind
```

A.3.5 /etc/exports

```
# /tftpboot/kickstart/ *(ro)
/tftpboot 10.162.xxx.56(ro)
```

A.3.6 /etc/xinetd.d/tftp

```
# default: off
# description: The tftp server serves files using the trivial file transfer \
# protocol. The tftp protocol is often used to boot diskless \
# workstations, download configuration files to network-aware printers, \
# and to start the installation process for some operating systems.
service tftp
{
    socket_type = dgram
    protocol = udp
    wait = yes
    user = xxxxx
    server = /usr/sbin/in.tftpd
    server_args = -s /tftpboot -v
    disable = no
    per_source = 11
    cps = 100 2
    flags = IPv4
}
```

A.3.7 /etc/mirror.defaults

```
package=defaults
# The LOCAL hostname - if not the same as 'hostname'
# (I advertise the name sunsite.org.uk but the machine is
# really swallow.sunsite.org.uk.)
hostname=praktikant-51.xxx.xxx.xxx.x.net
# Keep all local_dirs relative to here
local_dir=/data/
# The local_dir must exist FIRST
#local_dir_check=true
remote_password=malte@gmx.de
mail_to=root
# Don't mirror file modes. Set all dirs/files to these
dir_mode=0755
file_mode=0444
# By defaults files are owned by root.zero
user=0
group=0
# Keep a log file in each updated directory
```

```

update_log=.mirror
# update_log=
# Don't overwrite my mirror log with the remote one.
# Don't pull back any of their mirror temporary files.
# nor any FSP or gopher files...
exclude_patt=(^/)(\.mirror$\|.mirror\.log|core$\|.cap$\.in\.*\.$|MIRROR\|LOG|#.*#\|FSP$\|.cache$\|.zipped$\|.notar$\|.message|lost\+found\|Network
Tras h Folder)|suky.mpe?g
# Do not to compress anything compress_patt= compress_prog=compress
# Don't compress information files, files that don't benifit from
# being compressed, files that tell ftpd, gopher, wais... to do things,
# the sources for compression programs...
# (Note this is the only regexp that is case insensitive.)
# z matches compress/pack/gzip, gz for gzip. (built into perl)
# taz/tgz is compressed or gzipped tar files
# arc, arj, lzh, zip and zoo are pc and/or amiga archives.
# sea are mac archives.
# vms used -z instead of .z. stupid vms.
# shk is multimedia? used on apple2s.
# rpm and deb are package formats used on RedHat and Debian Linux
#compress_excl+|-z(\d+)?$\|.tgz\|_tgz\|.tar\|.Z\|.tar\|.gz\|.taz$\|.arc$\|. zip$\|.lzh$\|.zoo$\|.exe$\|.lha$\|.zom$\|.gif$\|.jpeg$\|.jpg$\|.mpeg$\|.au$\|.s
hk$|rpm$|deb$|read.*me|index|info|faq|gzip|compress|^/\|)\.?$
# Don't delete own mirror log, .notar or .cache files (incl in subdirs)
# delete_excl=(^/)(\.mirror|notar|cache)$
# Ignore any local readme and .mirror files local_ignore=README.doc.ic|^/\|)\.(mirror|notar)$
# Automatically delete local copies of files that the
# remote site has zapped
do_deletes=true
max_delete_files=50%
max_delete_dirs=50%
timeout=300 #failed_gets_excl=:\ Permission denied\.$
package=RPMS
comment=RedHat9 RPMS
site=ftp.rrz.uni-koeln.de
remote_dir=redhat/linux/9/en/os/i386/RedHat/RPMS/
# Local_dir+ causes gnu to be appended to the default local_dir
# so making /public/gnu local_dir+updates/9/i386/
# exclude_patt+|^ListArchives/|^lost+found/|^scheme-7.0/|^\.history
# I tend to only keep the latest couple of versions of things
# this stops mirror from retrieving the older versions I've removed
# max_days=100
do_deletes=false
# algorithm=1
passive_ftp=true

```

A.3.8 /etc/logrotate.conf

```

# see "man logrotate" for details
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# create new (empty) log files after rotating old ones
create
# uncomment this if you want your log files compressed
compress
# RPM packages drop log rotation information into this directory
include /etc/logrotate.d
# no packages own wtmp – we'll rotate them here

```

```

/var/log/wtmp
{
monthly
create 0664 root utmp
rotate 1
}

```

A.4 Bootloader und Skripte

A.4.1 Installations-PXE-Bootloader-Konfiguration

```

label linux
kernel vmlinuz
append initrd=initrd.img ks=nfs:10.162.xxx.55:/tftpboot/kickstart/ks.cfg ksdevice=eth0

```

A.4.2 default PXE-Bootloader-Konfiguration

```

default linux
label linux
localboot 0

```

A.4.3 PXEcfg_changer.pl

```

#!/usr/bin/perl -w
#
# Author: Alf Wachsmann, <alfw@slac.stanford.edu>, August 15, 2002
# configured for Colt telecom GmbH by malte eismann, <malte@fuldastrasse.de>, 12.11.2003
# Name of script: PXEcfg_changer.pl
# Version 2.0
# usage:
# $path/PXEcfg_changer.pl
use strict;
use vars qw($VERSION $PRG $PATH);
my $debug = 0;
my $verbose = 1;
my $file = '/var/log/messages';
$VERSION = '2.0';
$PRG = $0;
$PRG =~ s{(.*/)}{};
$PATH = $1;
my $PXEtftpPATH = "/tftpboot/pxelinux.cfg";
my $PXEdefault = $PXEtftpPATH."/install";
die("Directory '$PXEtftpPATH' does not exist!") unless (-d $PXEtftpPATH);
die("File '$PXEdefault' does not exist!") unless (-f $PXEdefault);
my $hexIP_prefix = uc(sprintf("%02x", 10).sprintf("%02x", 162));
my $pid = open(INPUT, "/usr/bin/tail -n 1 -f $file |");
$SIG{INT} = $SIG{TERM} = sub { system("kill $pid") };
while (<INPUT>) {
# this regexp matches output from tftpd that looks like this:
# tftpd[9960]:RRQ from 10.162.xxx.56 filename kickstart_end
next unless $_ =~ /tftpd\[d+\]: RRQ from 10\.\d2\.\(d+\)\.\(d+\) filename kickstart_end/;
my $hexIP = uc(sprintf("%02x", $1).sprintf("%02x", $2));
print "unlink($PXEtftpPATH/$hexIP_prefix$hexIP)\n" if $debug|$verbose;
unlink($PXEtftpPATH."/".$hexIP_prefix.$hexIP) unless $debug;
}

```

A.4.4 gethostip.c

```
#ident "$Id: projekt.lyx,v 1.1.1.1 2004/01/14 11:28:42 malte Exp $"
/* _____ */
*
* Copyright 2001 H. Peter Anvin - All Rights Reserved
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, Inc., 675 Mass Ave, Cambridge MA 02139,
* USA; either version 2 of the License, or (at your option) any later
* version; incorporated herein by reference.
*
* _____ */
/*
* gethostip.c
*
* Small program to use gethostbyname() to print out a hostname in
* hex and/or dotted-quad notation
*/
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <sys/socket.h>
#include <unistd.h>
#include <sys/exits.h>
#define _GNU_SOURCE /* For getopt_long */
#include <getopt.h>
const struct option options[] =
{
  { "hexadecimal", 0, NULL, 'x' },
  { "decimal", 0, NULL, 'd' },
  { "dotted-quad", 0, NULL, 'd' },
  { "full-output", 0, NULL, 'f' },
  { "name", 0, NULL, 'n' },
  { "help", 0, NULL, 'h' },
  { NULL, 0, NULL, 0 }
};
const char *program;
void usage(int exit_code)
{
  fprintf(stderr, "Usage: %s [-dxnf] hostname/ip...\n", program);
  exit(exit_code);
}
int main(int argc, char *argv[])
{
  int opt;
  int output = 0;
  int i;
  char *sep;
  char dotquad[16];
  int err = 0;
  struct hostent *host;
  program = argv[0];
  while ( (opt = getopt_long(argc, argv, "dxfnh", options, NULL)) != -1 ) {
    switch ( opt ) {
      case 'd':
        output |= 2; /* Decimal output */
        break;

```

```

case 'x':
output |= 4; /* Hexadecimal output */
break;
case 'n':
output |= 1; /* Canonical name output */
break;
case 'f':
output = 7; /* Full output */
break;
case 'h':
usage(0);
break;
default:
usage(EX_USAGE);
break;
}
}
if ( optind == argc )
usage(EX_USAGE);
if ( output == 0 )
output = 7; /* Default output */
for ( i = optind ; i < argc ; i++ ) {
sep = "";
host = gethostbyname(argv[i]);
if ( !host ) {
herror(argv[i]);
err = 1;
continue;
}
if ( host->h_addrtype != AF_INET || host->h_length != 4 ) {
fprintf(stderr, "%s: No IPv4 address associated with name\n", argv[i]);
err = 1;
continue;
}
if ( output & 1 ) {
printf("%s%s", sep, host->h_name);
sep = " ";
}
if ( output & 2 ) {
sprintf(dotquad, "%u.%u.%u.%u",
((unsigned char *)host->h_addr)[0],
((unsigned char *)host->h_addr)[1],
((unsigned char *)host->h_addr)[2],
((unsigned char *)host->h_addr)[3]);
printf("%s%s", sep, dotquad);
sep = " ";
}
if ( output & 4 ) {
unsigned long addr =
(((unsigned char *)host->h_addr)[0] << 24UL) +
(((unsigned char *)host->h_addr)[1] << 16UL) +
(((unsigned char *)host->h_addr)[2] << 8UL) +
(((unsigned char *)host->h_addr)[3]);
printf("%s%08lX", sep, addr);
sep = " ";
}
}
putchar('\n');
}

```

```
return err;
}
```

A.4.5 update.sh

```
#!/bin/bash
umask 022
echo "mirrorin..."
/usr/bin/mirror
echo "merging..."
# merging
# the i386 part :-)
# The following will update rpms in a RedHat distribution found in $RPMDIR.
# The old rpms will be placed in $OLDDIR.
# The new rpms should be located in $UPDDIR.
# The new images are in $IMGDIR
# The images to be updated are in $OMGDIR
SYSTEMS="i386 i686"
REDHATS="9"
for RHVS in $REDHATS ; do
echo "working in $RHVS..."
for SYST in $SYSTEMS ; do
RHROOT=/data/RedHat${RHVS}
RPMDIR=${RHROOT}/RedHat/RPMS
UPDDIR=/data/updates/${RHVS}/${SYST}
OLDDIR=${RHROOT}/old_${SYST}
if [ ! -d $OLDDIR ] ; then
echo making directory $OLDDIR
mkdir $OLDDIR
fi
allow_null_glob_expansion=1
for rpm in `ls ${UPDDIR}/*.${SYST}.rpm` ; do
NAME=`rpm -queryformat "%{NAME}" -qp $rpm`
# echo "NAME: $NAME"
unset OLDNAME
for oldrpm in `ls ${RPMDIR}/${NAME}.rpm` ; do
Q1=`rpm -queryformat "%{NAME}" -qp $oldrpm`
# echo "Q1: $Q1 = $NAME ?"
if [ `rpm -queryformat "%{NAME}" -qp $oldrpm` == "$NAME" ] ; then
OLDNAME=$oldrpm;
# echo "send BREAK!!!!!"
break
fi
done
if [ -z "$OLDNAME" ] ; then
echo $NAME is new
cp -pv $rpm $RPMDIR
else
if [ `basename $rpm` != `basename $OLDNAME` ] ; then
mv $OLDNAME $OLDDIR
cp -pv $rpm $RPMDIR
fi
fi
done
done
echo generating hdlist...
echo "genhdlist for RedHat${RHVS}"
```

```
/usr/lib/anaconda-runtime/genhdlst --withnumbers --hdlst /data/RedHat${RHVS}/RedHat/base/hdlst /data/RedHat${RHVS}/  
|| echo "**** GENHDLIST FAILED ****"  
done
```

A.5 Logfiles

A.5.1 Auszug Installation /root/install.log

```
Installing 352 packages  
-snip-  
Installing glibc-common-2.3.2-11.9.  
Installing hwdata-0.75-1.  
Installing redhat-logos-1.1.12-1.  
Installing setup-2.5.25-1.  
Installing filesystem-2.2.1-3.  
Installing basesystem-8.0-2.  
Installing glibc-2.3.2-11.9.  
Installing bzip2-libs-1.0.2-8.  
Installing chkconfig-1.3.8-1.  
Installing cracklib-2.7-21.  
Installing db4-4.0.14-20.  
Installing e2fsprogs-1.32-6.  
Installing elfutils-libelf-0.76-3.  
Installing expat-1.95.5-2.  
Installing gdbm-1.8.0-20.  
Installing glib-1.2.10-10.  
Installing glib2-2.2.1-1.  
Installing gmp-4.1.2-2.  
Installing hdparm-5.2-4.  
Installing iputils-20020927-2.  
Installing libattr-2.2.0-1.  
Installing libacl-2.2.3-1.  
Installing libgcc-3.2.2-5.  
Installing losetup-2.11y-9.  
Installing mingetty-1.01-1.  
Installing mktemp-1.5-18.  
Installing mount-2.11y-9.  
Installing net-tools-1.60-12.  
Installing pcre-3.9-10.  
Installing popt-1.8-0.69.  
Installing setserial-2.17-12.  
Installing shadow-utils-4.0.3-6.  
Installing slang-1.4.5-16.  
Installing newt-0.51.4-1.  
Installing termcap-11.0.1-16.  
Installing libtermcap-2.0.8-35.  
Installing bash-2.05b-20.  
Installing iproute-2.4.7-7.  
Installing lvm-1.0.3-12.  
Installing MAKEDEV-3.3.2-5.  
Installing ncurses-5.3-4.  
Installing lsof-4.63-4.  
Installing mailcap-2.1.13-1.  
Installing mailx-8.1.1-28.  
Installing make-3.79.1-17.  
Installing man-1.5k-6.  
Installing man-pages-1.53-3.  
Installing minicom-2.00.0-12.
```

Installing rdist-6.1.5-26.
Installing telnet-0.17-25.
Installing time-1.7-21.
Installing tmpwatch-2.8.4-5.
Installing traceroute-1.4a12-9.
Installing unix2dos-2.2-19.
Installing unzip-5.50-7.
Installing up2date-3.1.23-1.
Installing utempter-0.5.2-16.
Installing vconfig-1.6-2.
Installing vixie-cron-3.0.1-74.
Installing anacron-2.3-25.
Installing wget-1.8.2-9.
Installing wvdial-1.53-9.
Installing ypbind-1.11-4.
Installing yp-tools-2.7-5.
Installing zip-2.3-16.
Installing gdk-pixbuf-0.18.0-7.
Installing libungif-4.1.0-15.
Installing ImageMagick-5.4.7-10.
Installing w3m-0.3.2.2-5.
Installing libgnomeui-2.2.0.1-5.
Installing pygtk2-1.99.14-4.
Installing gnome-python2-canvas-1.99.14-5.
Installing pygtk2-libglade-1.99.14-4.
Installing redhat-config-keyboard-1.0.3-4.
Installing redhat-config-kickstart-2.3.6-4.
Installing redhat-config-language-1.0.4-1.
Installing redhat-config-packages-1.1.8-1.
Installing redhat-config-proc-0.21-1.
-snap-

A.5.2 Auszug .mirror-Logfile

-snip-
mirroring RPMS (kickstartserver:/pub/RedHat/RPMS) completed successfully @ 10 Dec 103 13:15
Got mutt-1.5.5.1-1.i386.rpm 1323803 0
mirroring RPMS (kickstartserver:/pub/RedHat/RPMS) completed successfully @ 10 Dec 103 13:58
mirroring RPMS (ftp.rz.uni-koeln.de:redhat/linux/9/en/os/i386/RedHat/RPMS/) completed successfully @ 10 Dec 103 14:34
Got tree-1.2-22.i386.rpm 14732 0
-snap-