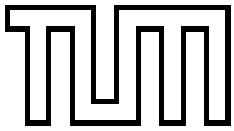


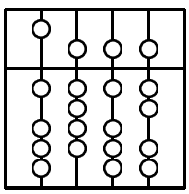
INSTITUT FÜR INFORMATIK

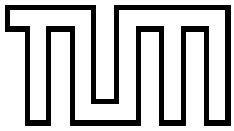
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Systementwicklungsprojekt

Konzeption und Realisierung einer sicheren Multiboot- und Backup-Infrastruktur

Bearbeiter:	Alexander Ilic, Christian Lieb
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Helmut Reiser Markus Garschhammer





INSTITUT FÜR INFORMATIK

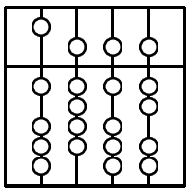
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Systementwicklungsprojekt

Konzeption und Realisierung einer sicheren Multiboot- und Backup-Infrastruktur

Bearbeiter:	Alexander Ilic, Christian Lieb
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Helmut Reiser Markus Garschhammer

Abgabetermin: 12. Februar 2003



Zusammenfassung

In diesem Systementwicklungsprojektwird eine sichere Multiboot- und Backupinfrastruktur für die am Lehrstuhl durchgeführten Praktika, das Rechnernetzpraktikum und das IT-Sicherheitspraktikum, entworfen und realisiert. Diese Infrastruktur ist im Sinne der Mandantenfähigkeit leicht erweiterbar, bietet den Benutzern Sicherheit durch Abschotten der Mandanten, Verschlüsselung des Datentransfers und ein einfach zu bedienendes Backupsystem.

Inhaltsverzeichnis

Inhaltsverzeichnis i

Abbildungsverzeichnis iii

Tabellenverzeichnis iv

1 Einleitung 1
- 1.1 Motivation . . . 1
- 1.2 Aufgabenstellung . . . 1
- 1.3 Inhaltsüberblick . . . 2

2 Designkriterien 3
- 2.1 Zentralisierung . . . 3
- 2.2 Sicherheit . . . 3
- 2.3 Einfache Bedienung . . . 4
- 2.4 Skalierbarkeit . . . 4

3 Realisierung 5
- 3.1 Kryptofilesystem . . . 5
  - 3.1.1 Auswahl des Kryptofilesystems . . . 5
  - 3.1.2 Partitionskey vs. Partitionspasswort . . . 6
  - 3.1.3 Booten von Kryptopartitionen . . . 7
- 3.2 Remote-Bootmanager . . . 8
  - 3.2.1 BPBatch . . . 8
  - 3.2.2 Aufbau des Bootmenüs . . . 9
  - 3.2.3 Skalierbarkeit . . . 10
  - 3.2.4 Fallback System . . . 10
  - 3.2.5 Booten von Images . . . 10
- 3.3 Backuplösung . . . 10
  - 3.3.1 Vergleich verschiedener Backuplösungen . . . 11
  - 3.3.2 Verwendete Backuplösung . . . 12
  - 3.3.3 Evaluation . . . 13

4 Benutzungsanleitung 15
- 4.1 Einrichten des Servers . . . 15
- 4.2 Installation der Clients . . . 16
- 4.3 Erzeugen von Images . . . 16
  - 4.3.1 Windowsimage . . . 16
  - 4.3.2 Linuximage . . . 17
- 4.4 Update eines Betriebsystems . . . 17
  - 4.4.1 Update von Suse 8.0 auf Suse 9.0 . . . 17
  - 4.4.2 Umstellen auf andere Betriebsysteme . . . 18

4.5 SECP Backup/Restore Befehle 18
4.5.1 listbackups 18
4.5.2 dobackup 18
4.5.3 dorestore 18
4.5.4 Beispiele 18
4.5.5 Anmerkungen 19
4.6 Kryptographisches Filesystem 19
4.7 Änderung der Partitionierung 19

5 Zusammenfassung **20**

A Quelltexte **21**

B Konfigurationsdateien **35**

Literaturverzeichnis **41**

Abbildungsverzeichnis

3.1 Partitionierung 7
3.2 Bootvorgang - Schematisch 9
3.3 Funktionsweise Schematisch 13
3.4 Anzahl der durchgeführten Backups 13
3.5 Platzbedarf der Slots 14
3.6 Platzbedarf Gesamt 14

Tabellenverzeichnis

3.1 Vergleich von Backuplösungen nach technischen Aspekten 11
3.2 Vergleich von Backuplösungen nach administrativen Aspekten 12

Kapitel 1

Einleitung

Im Verlaufe des Systementwicklungsprojektes wurde eine sichere Boot- und Backup-Infrastruktur entworfen und realisiert. In der folgendene Einleitung wird ein Einblick in die Motivation und die damit verbundene Aufgabenstellung des Systementwicklungspraktikums geboten.

1.1 Motivation

Seit dem Wintersemester 2002/2003 finden am Lehrstuhl Hegering das IT-Sicherheits- und das Rechnernetzpraktikum in den selben Praktikumsräumen statt. Den Teilnehmern stehen dort zehn Rechner gleicher Hardware zur Verfügung, an denen Sie Ihre Aufgaben bearbeiten können. Auf Grund hoher Teilnehmerzahlen ist es erforderlich, dass Teilnehmergruppen aus beiden Praktika zeitversetzt an einem Rechner arbeiten müssen. Zusätzlich sollen jeweils zwei Gruppen aus dem IT-Sicherheitspraktikum an dem selben Rechner arbeiten können. Fast unumgänglich traten hier in der Vergangenheit Probleme auf: Teilnehmer der vorherigen Gruppe vergaßen beispielsweise, selbst umgestellte Konfigurationen rückgängig zu machen. So hatten die Nachfolgergruppen durchaus öfters eine erhöhte Arbeitszeit, da die Rechner zunächst wieder einmal in die Ausgangskonfiguration gebracht werden mussten. Aber auch das Gegenteil war manchmal der Fall: Fertige Lösungen der vorherigen Gruppe waren auf den Rechnern zu finden. Um sicherzustellen, dass der Arbeitsaufwand aller Teilnehmer bei der Bearbeitung ihres Praktikums nahezu gleich ist, empfiehlt sich eine Neugestaltung der bisher verwendeten Infrastruktur.

1.2 Aufgabenstellung

Im Systementwicklungsprojekt soll eine sichere Boot- und Backup-Infrastuktur entworfen und realisiert werden, die in den Praktika zum Einsatz kommen soll. Den Benutzern soll ein komfortables Bootmenü zur Verfügung gestellt werden, bei dem sie das von ihnen benötigte Betriebsystem booten können. Im IT-Sicherheitspraktikum sollen zeitversetzt zwei Gruppen am selben Rechner arbeiten können, was zwei Instanzen des Betriebsystems erfordert. Beide Gruppen sollen mit root-Rechten in einem Linuxbetriebsystem arbeiten können. Für das Rechnernetzpraktikum wird sowohl ein Linux, als auch ein Windowsbetriebsystem benötigt. Auch das Rechnernetzpraktikum findet an den selben Praktikumsrechnern statt. Wohingegen Änderungen im Rechnernetzpraktikum nicht abgespeichert werden müssen, ist dieses beim IT-Sicherheitspraktikum erforderlich. In der Infrastruktur soll es nicht möglich sein, die Daten anderer Gruppen einzusehen oder auf diese Daten zuzugreifen. Den Teilnehmern des IT-Sicherheitspraktikums soll die Möglichkeit gegeben sein, selbständig wichtige Daten zu sichern und wiederherstellen zu können. Ist es den Teilnehmern aus irgdenwelchen Gründen nicht möglich, selbständig ihre Daten wiederherzustellen, so soll der Praktikumbetreuer Möglichkeiten besitzen, diese Aufgabe zu übernehmen.

1.3 Inhaltsüberblick

Anhand der Anforderungen wird im Verlauf der Ausarbeitung ein Design entworfen, getestet und realisiert. In einer Benutzeranleitung werden konkret die durchzuführenden Schritte für eine Installation der Infrastruktur erläutert. Wichtige Punkte, wie das Erstellen von Images oder das manuelle anpassen von Konfigurationsdateien, und oft in der Praxis auftretende Probleme werden hervorgehoben. Schließlich folgt eine Zusammenfassung und ein Ausblick des Systementwicklungsprojektes, indem Erweiterungsmöglichkeiten der Infrastruktur angesprochen werden.

Kapitel 2

Designkriterien

Um der Aufgabenstellung gerecht zu werden, wurden in der Konzeptionsphase die Designkriterien für die Infrastruktur festgelegt.

2.1 Zentralisierung

Die Verwendung des Praktikumsraumes kann sich je nach Inhalt der dort abgehaltenen Praktika ändern. Ebenso kann nicht vorausgesetzt werden, dass die Rechner Anzahl und die eingesetzten Betriebssysteme auf Dauer gleich bleiben. Eine einfache zentrale Anpassung soll die Wartbarkeit deshalb deutlich erhöhen.

Das komplette Bootkonzept ist netzwerkgestützt und in Abhängigkeit eines Bootservers erstellt worden. Somit lassen sich Änderungen im Bootmenü für alle Rechner, durch modifizieren einer Konfigurationsdatei am Server global und zeitnah umsetzen. Ebenso wurde die Backuplösung so konzipiert, dass die Logik und Konfiguration auf einem Backupserver liegt. Die Integration neuer Clients ist deshalb sehr leicht möglich.

2.2 Sicherheit

Es wurde gefordert, dass die Besitzer(Benutzer) der einzelnen lokal installierten Betriebssysteme sich nicht gegenseitig Ausspähen können. Es wird angenommen, dass die Benutzer für das Absichern gegen Angriffe aus dem Netz selber verantwortlich sind, da sie auch bestimmen welche Dienste bei ihnen zugänglich sind und welche Software installiert ist. Um dem Ausspähen von Daten durch mounten von fremden lokalen Filesystemen entgegen zu wirken, ist der Einsatz eines Kryptofilesystems unerlässlich. Das heißt, alle sensitiven Daten liegen nur in verschlüsselter Form auf dem Datenträger vor. Die Daten sind über eine, vom Benutzer änderbare, Passphrase geschützt. Diese muss beim Booten des Systems eingegeben werden. Da die Rechner auch über das Backupsystem gesichert werden sollen, ist außerdem eine Masterpassphrase erforderlich, da sonst nur Backups durch den Benutzer durchgeführt werden können und nicht durch einen Administrator. Die Daten dürfen beim Backup dann natürlich nicht unverschlüsselt über das Netz übertragen werden, sonst wären die sensitiven Daten einem Packetsniffer gegenüber transparent mitlesbar. Deshalb wird gefordert, dass alle über das Netz gesendeten und empfangenen Daten, in Zusammenhang mit der Backuplösung, verschlüsselt sein müssen.

2.3 Einfache Bedienung

Für den täglichen Einsatz und dem Gewährleisten einer möglichst ausfallresistenten Lösung, ist die kontinuierliche Sicherung der Benutzerdaten durch das Backupsystem unerlässlich. Deshalb muss das System sehr einfach durch den Benutzer anstoßbar und verwendbar sein. Auch vom Kryptofilesystem sollte der Benutzer nicht behindert werden, sondern das Ganze sollte transparent ablaufen. Ein regelmäßiges Ändern der Partitionpassphrase wäre wünschenswert. Daher muss der Benutzer selbstständig und ohne Aufwand vollziehen können.

2.4 Skalierbarkeit

Sowohl die Backup/Restore Software, als auch der Bootmanager müssen sich als performant genug erweisen um einen Einsatz in größeren Netzen zu ermöglichen. Die Infrastruktur darf keine Einschränkung in der Anzahl der Rechner und konfigurierten Betriebssysteme stellen.

Kapitel 3

Realisierung

Die Beschreibung der Lösung gliedert sich in drei Teile:
Im ersten Teil wird das Kryptofilesystems vorgestellt, im zweiten Teil wird der verwendete Bootmanager `BpBatch` erklärt, und im dritten Teil wird genauer auf die realisierte Backuplösung eingegangen.

3.1 Kryptofilesystem

Durch den Einsatz des Kryptofilesystems wird sichergestellt, dass Daten auf der Festplatte nur in verschlüsseltem Zustand gespeichert werden. Im folgenden wird die Auswahl des verwendeten Kryptofilesystems, sowie das Design der Partitionstabelle erläutert. Des weiteren wird die Realisierung und eine abschließende Performance und Sicherheitsdiskussion durchgeführt um Rückschlüsse auf mögliche Nachteile des Einsatzes eines Kryptofilesystems zu schließen.

3.1.1 Auswahl des Kryptofilesystems

Die Anforderung war nach einer transparenten Verschlüsselung, so dass der Benutzer im Wesentlichen nicht durch das Kryptofilesystem beeinträchtigt wird oder es im optimalen Fall nicht einmal bemerkt. Betrachtet wurden drei verschiedene Opensource Kryptofilesysteme die der Forderung nach Transparenz genügen: TCFS, PPDD und die Linux CryptoAPI. Diese drei Systeme werden im folgenden knapp vorgestellt und danach miteinander verglichen.

TCFS

TCFS arbeitet mit einer Kombination aus einem Kernelmodul und einem modifizierten NFS Daemon, der für die Verschlüsselung/Entschlüsselung zuständig ist. Unterstützt werden die Algorithmen CBC-DES, IDEA und RC5. TCFS konnte aber leider nicht mal testweise zum Einsatz gebracht werden, da die letze Version des Kernelmoduls nur für Kernel 2.0.X erhältlich ist. Die Entwicklung scheint zwar dennoch im Gange zu sein, aber TCFS scheint mehr als Ablösung für NFS gedacht zu sein, als für ein lokales Verschlüsselungssystem.

PPDD

PPDD ist ein device Treiber, der als Kernelmodul geladen wird und das loopback device benutzt. Als Verschlüsselungsalgorithmus steht lediglich Blowfish zur Verfügung; die Schlüssellänge ist hierbei leider

nicht einstellbar. Dagegen überzeugt PPDD bei dem Managen von Passphrases. So gibt es eine Master- und eine Workpassphrase, die nach belieben geändert werden kann. Ebenso ist es möglich, ein Filesystem einfach per Befehl vom Verschlüsselten Zustand in ein Nichtverschlüsseltes zurück zu konvertieren. Ein Nachteil von PPDD ist aber, daß die Kernelpatches nicht immer erfolgreich anwendbar sind und somit ein beliebiges Upgraden des Kernels erst bei Vorhandensein eines passenden PPDD Patches möglich ist.

CryptoAPI

Die Kernel CryptoAPI ist aus historischen Gründen noch nicht direkt im Kernel enthalten (Kryptoderegulierung einiger Länder). Sie wird jedoch parallel zum Kernelsource weiterentwickelt und somit passen die Patches der CryptoAPI immer direkt zum aktuellen Kernelsource. Die CryptoAPI stellt ein modulares Verschlüsselungssystem zur Verfügung, so kann z.B. aus verschiedenen vorhanden Algorithmen ausgewählt, oder relativ einfach neue hinzugefügt werden. Das Loopback device wird benutzt, um die Verschlüsselung/Entschlüsselung zu übernehmen. Das Filesytem wird verschlüsselt über genau einen Key auf dem physischen Medium abgelegt und dieser kann nicht geändert werden.

Auswahl eines Systems

Die Auswahl fiel auf die CryptoAPI, da diese parallel zum Kernel entwickelt wird und somit ein Upgraden sehr einfach ist. Die im Praktikum verwendeten Rechner setzen dazu noch SuSE als Linux Distribution ein, die die CryptoAPI gleich beinhaltet. Als Verschlüsselungsalgorithmus wird das twofish Modul mitgeliefert und deshalb wurde dieses verwendet. Trotzdem wäre es ohne Probleme möglich, ein anderes Modul wie z.B. AES zu benutzten. PPDD hat zwar enorme Vorteile im Bereich Keymanagement, aber durch die Upgradefähigkeit konnte nur die CryptoAPI bestechen. TCFS kam, wie schon oben erwähnt, nicht in die engere Auswahl, da es nicht getestet werden konnten.

3.1.2 Partitionskey vs. Partitionspasswort

Eine Anforderung an die Infrastruktur war, dass sie leicht administrierbar sein muss. Zum Booten eines Systems auf einem Kryptofilesystem, muss der Benutzer ein Passwort kennen. Es kommt nicht selten vor, dass ein Benutzer sein Passwort vergisst. Ist nun die komplette Partition mit diesem Passwort verschlüsselt gewesen, so kann man nur durch einen Bruteforce Angriff wieder an die Daten kommen.
Man hat daher zwei Anforderungen bezüglich Passwörter und Kryptofilesystem. Die erste ist, dass es ein Masterpasswort gibt. Mit diesem kann die Partition dann auch noch in Notfällen gebootet oder gesichert werden. Die zweite Forderung resultiert aus der ersten und zwar sollte es nach einem Booten mit dem Masterpasswort möglich sein, dem Benutzer ein neues Passwort für sein Filesystem zu setzen.
Leider bieten bis auf PPDD kein Kryptofilesystem die direkte Umsetzung dieser Anforderung. In diesem Systementwicklungsprojektwurde jedoch eine Möglichkeit erarbeitet, diese Anforderung trotzdem in Zusammenhang mit der CryptoAPI umzusetzen. Dazu wird zunächst ein möglichst zufälliger und langer Key generiert, mit dessen Hilfe die komplette Partition verschlüsselt wird. Dieser Key wird per gpg symmetrisch verschlüsselt und in einer Datei auf der Kryptoboot Partition in einem, dediziert für die verschiedenen IT-Sicherheitsgruppen erstelltem, Verzeichnis abgelegt. In diesen Verzeichnissen befinden sich zwei Dateien, die den gleichen Key enthalten. Die Datei master.key ist mit der Masterpassphrase über den Algorithmus Blowfish verschlüsselt und die Datei user.key ist mit einer initialen Userpassphrase über Cast5 verschlüsselt. Der Benutzer muss diese Passphrase sofort beim ersten Booten seines Systems ändern. Es wurden verschiedene Algorithmen zum Verschlüsseln genommen, um die Komplexität eines Bruteforce Angriffs maximal zu halten. Ansonsten wäre es theoretisch möglich diese herabzusetzen, da zwei verschiedene Passphrases mit dem gleichen Algorithmus den gleichen Plaintext verschlüsseln würden.
In diesem Systementwicklungsprojektwurden sowohl die Methoden zur Generierung des Partitionskeys und das Einrichten der Kryptofilesysteme, als auch ein kleines Shellscript zum Ändern einer Benutzer-Passphrase implementiert. Die Quellcodes dazu befinden sich im Anhang (A).

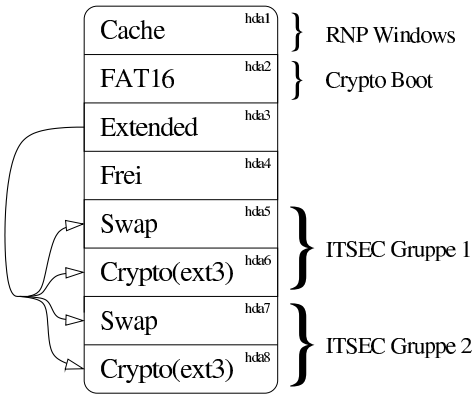


Abbildung 3.1: Partitionierung

3.1.3 Booten von Kryptopartitionen

Um von einem Kryptofilesystem booten zu können, muss man ein System mit unverschlüsseltem Dateisystem eingerichtet haben. Dies ist die Partition `/dev/hda2`, die zum Booten von IT-Sicherheitsgruppe 1 und 2 benutzt wird. Auf dieser Partition befindet sich ein kleines System, das den Bootvorgang der Kryptopartition vorbereitet. Dies ist eine sogenannte Initial Ramdisk(Initrd).
Die Initrd ist ein kleines filebasiertes Loopback Dateisystem, das in eine Ramdisk geladen wird. Dies geschieht unmittelbar nach laden des Kernels, aber noch vor dem Starten des eigentlichem Betriebssystems.
In der Initrd ist ein kleines Skript, welches den Benutzer zum Eingeben der Passphrase auffordert. Danach wird geprüft, ob die eingegebene Passphrase zum Entschlüsseln der Datei master.key oder user.key benutzt werden kann. Ist das Entschlüsseln nicht erfolgreich, so wird der Benutzer erneut zur Eingabe aufgefordert. Wurde der Partitionskey dann entschlüsselt, dann wird über `losetup` das Loopback Device eingestellt (siehe A.1). Dies übernimmt mit Hilfe des Keys eine transparente Entschlüsselung der Partition. Das Loopdevice wird anschließend als Rootverzeichnis gemounted und das eigentliche System gebootet.

3.2 Remote-Bootmanager

Mit Hilfe eines Bootmanagers sollen die Teilnehmer der Praktika ihr benötigtes Betriebsystem auswählen und booten können. Ein einfacher Bootmanager wie Lilo würde dieser Anforderung gerecht werden, jedoch bringt ein Bootmanager, der lokal auf den einzelnen Rechnern installiert ist, einige Nachteile mit sich. Änderungen der Konfigurationsdateien des Bootmanagers müssten auf jedem Praktikumsrechner durchgeführt werden. Das bringt mit zunehmender Anzahl der Praktikumsrechner auch einen proportional wachsenden Aufwand für die Systemadministratoren mit sich. Deshalb erscheint die Verwendung eines Remote-Bootmanagers sinnvoll, bei dem der Aufwand durch eine zentralisierte Struktur unabhängig von der Anzahl der verwendeten Rechner ist.

3.2.1 BPBatch

Verwendet wird der Remote-Bootmanager BPBatch. BPBatch wird kostenlos im Internet unter der URL `http://www.bpbatch.org` für jeden angeboten. BPBatch bietet dem User neben einem graphischen Interface, ein User Authentifizierungssystem und auch die Möglichkeit, Images einer Festplatten-Partition zu erstellen und zu booten.

Benötigte Dienste

Für die Bootvorgang werden noch weitere Dienste benötigt. Zunächst muss auf dem Server ein `DHCP`-Server eingerichtet werden. `DHCP` (Dynamic Host Configuration Protocol) ermöglicht einem Computer, seine eigenen Netzwerk Parameter herauszufinden und erlaubt auch eine dynamische Adresszuweisung in einem TCP/IP Netzwerk. Ein Netzwerk, indem DHCP verwendet wird, kann mühelos umkonfiguriert, vergrößert oder verkleinert werden, ohne einzelne Computer direkt umzukonfigurieren. Das DHCP Protokoll wird im RFC 2131 genau erklärt. Des weiteren wird ein TFTP Server benötigt, der ebenfalls auf dem Server eingerichtet werden muss. Das TFTP Protokoll (Trivial File Transfer Protocol) dient zum Transport von Daten in einem Netzwerk. Der TFTP Server wird von BPBatch benötigt, um beispielsweise die Konfigurationsdatei oder Images vom Server auf die Clients zu kopieren.

Bootvorgang

Remote-Booting ist eingeteilt in mehrere Schritte:

1. Wird ein Client gestartet, übernimmt, nach den üblichen Standard-Checks des BIOS, das PXE (Preboot Execution Environment) Boot ROM Kontorolle über den Rechner. Die PXE Software ist auf der Netzewrkkarte integriert.
2. Das Boot ROM erteilt eine DHCP Anfrage an den Server, um eine IP-Adresse zugewiesen zu bekommen.
3. Wenn dem Server der anfragende Computer bekannt ist, also die MAC-Adresse des Clients in der dhcpd.conf eingetragen ist, sendet er eine DHCP Antwort mit grundlegenden Informationen, die die IP-Adresse, die Subnetmask und den Namen des zu ladenden Bootmanagers beinhalten.
4. Über das TFTP Protokoll wird der Batch Interpreter vom Server auf den Client kopiert. Der Bootmanager ist ein kleines Programm mit dem Namen BPBatch.
5. Der Batch Interpreter wird auf dem Client ausgeführt. Zu diesem Zeitpunkt ist kein Betriebsystem geladen, lediglich die PXE Software wird ausgeführt. Der Bootmanager befindet sich im Speicher des Clients.
6. Weitere Kommandozeilenparameter aus der DHCP Antwort und der Name der auszuführenden batch Datei werden vom batch Interpreter ausgelesen.

7. Der weitere Bootvorgang hängt von der vorher vom Systemadministrator angepassten batch File ab.

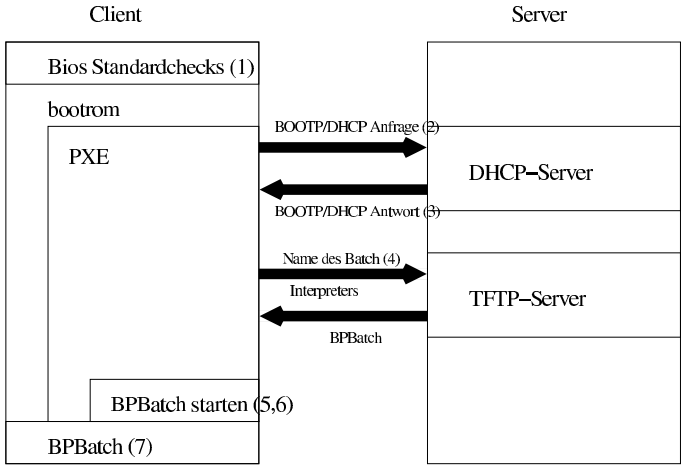


Abbildung 3.2: Bootvorgang - Schematisch

3.2.2 Aufbau des Bootmenüs

Die Gestaltung des Bootmenüs wird in dem Batchfile, `bpbatch.bpb`, festgelegt. Das Batchfile liegt standardmäßig in dem Verzeichnis `/tftpboot` auf dem Server. Der Ort des Batchfiles wird in der `dhcpd.conf` festgelegt. Grafische Optionen, wie die Auflösung, die Auswahl der Menüpunkte, Schriftart und Schrifgröße werden unter anderem in der Batchfile definiert. Insgesamt umfasst das Bootmenü sieben Punkte:

1. `RNP Linux`: Es wird eine Linux Suse Version 8.0 mit Hilfe eines Network File System auf dem Server gebootet.
2. `RNP Windows`: Das auf der ersten Partition liegende Windows wird gebootet. Dieses ist nur möglich wenn vorher einmal `RNP Windows Restore` ausgeführt wurde und damit das Windows Image vom Server auf den Client kopiert und entpackt wurde, anderenfalls meldet BPBatch beim booten einen Fehler.
3. `IT-Sicherheut Gruppe 1`: Eine Linux Suse Version 8.0 kann hier ebenfalls gebootet werden. Anders aber als das vorhergehenden Linux für das RNP Praktikum, befindet sich dieses Linux lokal auf einer Partition der Festplatte. Lediglich der Kernel und die Ramdiskfile werden bei jedem Start vom Server neu kopiert, denn falls einmal das Netzwerk ausfallen sollte, so kann mit Hilfe eines Fallback-Systems das Betriebsystem trotzdem noch gebootet werden
4. `IT-Sicherheit Gruppe 2` Der Bootvorgang läuft bei Gruppe 2 analog zum Bootvorgang der Gruppe 1 auf einer anderen Partition ab.
5. `Power Off` Der Rechner kann aus dem Bootmenü heraus ausgeschaltet werden.
6. `IT-Sicherheit Restore` Ein Restoresystem wird mit Hilfe eines Network File Systems auf dem Server gebootet.
7. `RNP Windows Restore` Das Windows Image wird neu vom Server auf den Client kopiert und entpackt. Nachdem das Windows Image erfolgreich entpackt wurde, wird die Registry und die autoexec.bat noch angepasst, damit beispielsweise der Computername eines Rechners in einem Netzwerk eindeutig ist und es hier nicht zu Konflikten kommt, falls zwei Clients in dem Netzwerk Windows booten. Dies geschieht mit dem BPBatch Befehl `Patch` in der bpbatch.bpb.

3.2.3 Skalierbarkeit

Das Startmenü ist leicht um neue Menüpunkte zu ergänzen. In der Konfigurationsdatei, `/sepfilebpbatch.bpb`, auf dem Server müssen nur einige kleine Änderungen vorgenommen werden, um auf allen Clients eine neue Bootoption im Menü vorzufinden. Ähnlich einfach ist es, neue Rechner in das Netzwerk zu integrieren. Die Konfigurationsdatei des DHCP Servers muss man hierzu nur ändern, wenn die IP-Adressen an die MAC Adressen der Netzwerkkarten fest gebunden sind, oder bei dynamischer Vergabe der IP-Adressen, die IP-Range erschöpft ist. Bei einem neuen Rechner sollte natürlich zunächst einmal der Restore Prozess angestoßen werden, um die Festplatte richtig zu partitionieren und für die IT-Sicherheitsgruppen das Default Image vorzuinstallieren.

3.2.4 Fallback System

Die Zentralisierung einer Infrastruktur bringt nicht nur Vorteile mit sich. So sind die Clients sehr abhängig vom Server und bei vielen Infrastrukturen bei dessen Ausfall unbrauchbar. Es wäre also schön wenn die Infrastruktur auch funktionieren würde, wenn der Server oder auch das Netzwerk ausfällt. Hierfür wurde ein Fallback System integriert. Schlägt das Booten von BPBatch fehl, da der Client keine Verbindung zum Server bekommt, wird ein lokaler Bootmanager, Lilo, gestartet. Es ist trotz Ausfall des Netzwerkes oder des Servers noch möglich, alle lokalen Betriebsysteme zu starten.

3.2.5 Booten von Images

Das Windows Betriebsystem für das Rechnernetzpraktikum wird von einem Image gebootet. Ein Image eines Betriebsystems ist eine einzelne Datei, die alle Daten und Verzeichnisse und ihre entsprechende Struktur einer bootfähigen Partition enthält.

Vorgang

Das verwendeter Windows-Image wurde mit Hilfe von MR.Zip, eines mit BPBatch mitgelieferten Programmes, von einem vorkonfigurierten und bootfähigen Windows-Betriebsystem erstellt und auf den Server kopiert. Temporäre Verzeichnisse werden beim Erstellen des Images ausgeschlossen. Ein Windows Image kann nicht vom aktuell laufendem Betriebsystem erstellt werden, da Windows den Zugriff auf gerade verwendete Dateien verweigert. Wichtig ist es, dass das Windows Image unter einem Betriebsystem erstellt wird erstellt wird, das die langen Dateinamen, die Windows verwendet, versteht. Also z.B. unter Suse8.0, Windows98 oder Windows 2000, nicht jedoch unter DOS. Das Windows Image steht nun den Clients zur Verfügung und wird beim Booten von `RNP Windows Restore` vom Server auf die erste Partition der Clients kopiert und entpackt. Die Partition ist nun bootfähig.

Performance

Das Image eines Windows´98 Betriebsystems ist ungefähr 300 Megabyte groß. Beim Restoren des Images, dauert das Booten knapp 5 Mintuten, da erst die gesamten 300MByte auf den Client kopiert werden müssen. Wird hingegen der Menüpunkt `RNP Windows` gestartet, so wird der Kopierprozess übersprungen und die Wartezeit beim Booten des Betriebsystems auf knapp 1 Minute gekürzt.

3.3 Backuplösung

Um den Anforderungen zu genügen wurden verschiedene Opensource Backuplösungen ausgewählt und getestet. Nach der Auswertung der Tests wurde eine Entscheidung für eine Backuplösung getroffen und

diese dann ins gesamt System integriert.

3.3.1 Vergleich verschiedener Backuplösungen

Zunächst wurden die Auswahlkriterien zusammengestellt (resultierend aus den Anforderungen Kapitel 2) und vielversprechende Kandidaten für die Tests ausgewählt. Der Vergleich der getesteten Lösungen ist in den folgenden Tabellen dargestellt. Die Auswahlkriterien teilen sich in zwei Gruppen, Administration und Technik, ein.

Technik

	afBackup	AMANDA	Bacula	kBackup	BackupPC
File pooling	nein	nein	nein	nein	ja
Komprimierung	ja	ja	ja	ja	ja
Inkrementelle Backups	ja	ja	ja	ja	ja
Backup-level konfigurierbar	nein	ja	nein	nein	nein
Basiert auf	-	dump/restore oder tar	-	tar/afio	tar
Verschlüsselte Übertragung	nein	nein	ja	nein	nein
Geschrieben in	C	C	C++	Shell script	Perl
Architektur	Client/Server	Client/Server	Client/Server	Standalone	Server

Tabelle 3.1: Vergleich von Backuplösungen nach technischen Aspekten

Administration

	afBackup	AMANDA	Bacula	kBackup	BackupPC
Unterstützte Dateisysteme	alle	ext2/3, SMB	alle	alle	alle
User/PW accounts	nein	nein	nein	nein	nein
Admin Interface	Web basiert	Kommandozeile	Kommandozeile	GUI	Web basiert
Backup Master account	nein	nein	nein	nein	nein
Einzelne Dateien wählbar	nein	nein	ja	ja	nein
Backup/Restore anstoßbar von	User	Admin/Cronjob	Admin/Cronjob	User	Admin/Cronjob

Tabelle 3.2: Vergleich von Backuplösungen nach administrativen Aspekten

3.3.2 Verwendete Backuplösung

Da keines der untersuchten Backuplösungen die gewünschte Funktionalität (siehe 3.3.1) bieten konnte, wurde, aufbauend auf den standard Backupwerkzeugen Dump/Restore, eine in Perl geschriebene Backuplösung entwickelt.

Funktionsprinzip

Die Idee war, ein sehr modulares, auf Standardkomponenten aufbauendes System, zu entwickeln, das allen Feature Anforderungen gerecht wird.
Auf dem Server befindet sich die komplette Anwendungslogik, so dass ein Softwareupgrade sehr schnell durchgeführt werden kann. Auf den Clients befinden sich lediglich kleine Wrapper Scripts, die Vorgänge auf dem Server anstossen können. Zu diesen Vorgängen zählen, das Anstossen von Backups, das Restoren von Daten und das Auflisten bereits gemachter Backups. Diese Scripts tragen analog zu Ihrer Funktion entsprechende Namen. Auf die Benutzung dieser Befehle wird in Kapitel 4 eingegangen. Diese sind dobackup, dorestore und listbackups. Sie befinden sich unter `/sbin` auf den Client Rechnern.
Da die komplette Anwendungslogik auf dem Server ist, muss der Server die Möglichkeit haben, auf dem Client Befehle auszuführen. Um dies zu ermöglichen wurde ssh mit Publickey Authentifizierung eingesetzt. Der entsprechende Publickey muß auf jedem Client installiert sein und ermöglicht so dann dem Server, das Ausführen von Befehlen ohne Eingabe eines Passwortes. SSH wurde auch noch aus einem weiteren Grund eingesetzt: Datensicherheit. Durch die SSH Tunnel wird gewährleistet, dass die komplette Kommunikation zwischen Clients und Server geschützt ist. Auf Abbildung 3.3 wird dieses Modell anschaulich dargestellt.

Authentifizierung Die verwendete Authentifizierung setzt sich prinzipiell aus einem dreier Tupel zusammen: Praktikumsgruppe, Computer und einem Passwort. Durch die topologische Anordnung begünstigt, verwendet das Backupsystem aber noch eine zweite zusätzliche Sicherheitsmassnahme, die Überprüfung der MAC Adresse des Rechners. Jeder Computer ist mit einer Netzwerkkarte direkt an dem Server angeschlossen und somit kann dieser direkt die MAC Adresse des zu authentifizierenden Clientrechners prüfen. Diese Option ist standardmässig aktiviert, kann aber auch bei Bedarf aus Topologischen Gründen deaktiviert werden.

Backupsystem Beim Backupvorgang führt der Server auf dem Clientrechner ein level-1 dump (d.h. Level 1 Dump + Standardimage = vollständiges Image) zum Standardimage durch und legt dies unter dem pool directory komprimiert auf dem Server ab. Die abgelegte Datei, die das Backup enthält, wird im folgenden als Slot bezeichnet. Über die config.pl am Server ist einstellbar, wie viele Slots jede Gruppe

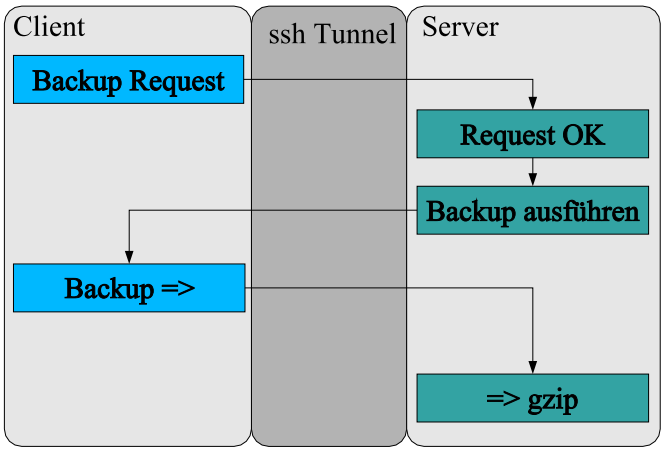


Abbildung 3.3: Funktionsweise Schematisch

nutzen darf, bevor ein alter Slot überschrieben werden muss. Über kleine Hilfsscripte auf der Clientseite (siehe 4 dobackup, dorestore, listbackup) hat der Benutzer aber immer im Überblick wie viele Slots noch frei sind oder kann auch dann entsprechend entscheiden, welchen Slot er überschreiben will, falls dies erforderlich ist.

3.3.3 Evaluation

Es wurden Tests auf Performance durchgeführt, sowie eine Analyse des Backupverhaltens der User im IT Sicherheitspraktikum Wintersemester 2002/2003.
Das Diagramm in Abbildung 3.4 zeigt die Anzahl der gemachten Backups pro Woche. Man erkennt an

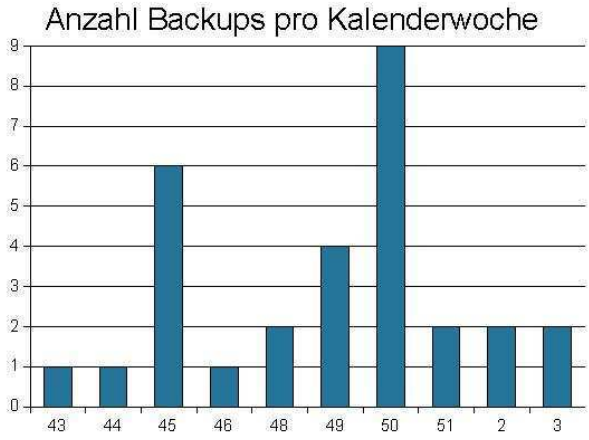


Abbildung 3.4: Anzahl der durchgeführten Backups

diesem Diagramm, dass die Benutzer erst nach und nach sich an das Backupsystem herangetraut haben. Und somit erst in Kalenderwoche 45 ernsthaft davon Gebrauch gemacht haben. Dies liegt vermutlich an der Einarbeitungszeit, die für Technik und Aufgaben aufgewendet werden musste, und somit wenig Zeit für die Einarbeitung in das Backupsystem blieb. Kurz vor Beginn der Weihnachtsfeiertage und nachdem der größte Teil der Aufgaben erledigt war, zeigt sich ein Peak bei Kalenderwoche 50, bei dem die Teilnehmer

ihre Aufgaben auf jeden Fall sicher haben wollten.

In Abbildung 3.5 sieht man, dass die Größe der Slots nicht bei allen Gruppen gleich ist, sondern von 7 MB bis zu 97 MB variiert. Die Diskrepanz zwischen minimaler und maximaler Slotgrösse hängt von den Be-

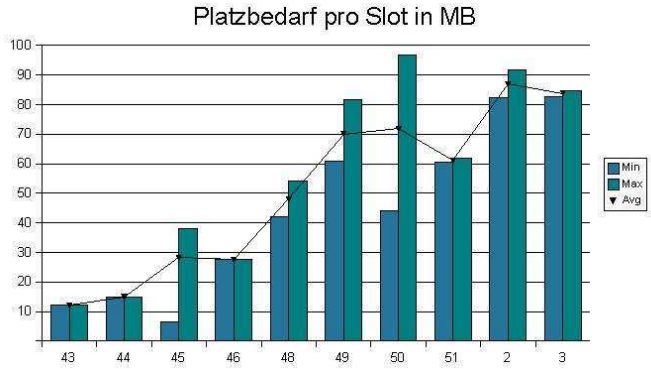


Abbildung 3.5: Platzbedarf der Slots

nutzern des jeweiligen Systems ab. Die Backups werden inkrementell Level-1 zum Defaultimage erstellt, was bedeutet, dass im Laufe der Zeit die Größe der Slots einer Gruppe monoton zunehmen. Die Zunahme der Größe ist Abhängig von den Veränderungen im Filesystem. Gruppen die viel Software installieren/-kompilieren besitzen somit größere Slots, als Gruppen, die weniger Veränderungen vollziehen.

Insgesamt ist der Platzverbrauch für die ganzen gesicherten Slots am Ende des Semesters 1.7 GB (siehe Abbildung 3.6) für 10 Rechner, die von 2 Gruppen genutzt werden. Man könnte noch etwas Platz sparen, indem man die Backups jedes mal inkrementell zum vorherigen Backup laufen lässt, aber man verliert dadurch den Vorteil, ein komplettes System einfach durch Defaultimage + beliebigen Level-1 dump wiederherstellen zu können.

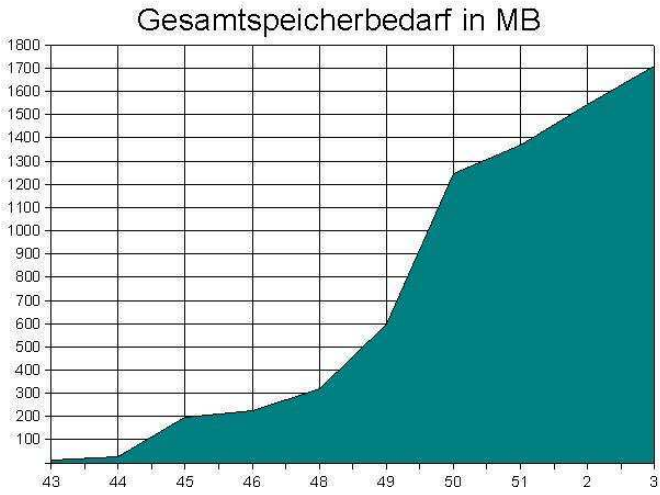


Abbildung 3.6: Platzbedarf Gesamt

Kapitel 4

Benutzungsanleitung

In diesem Abschnitt wird kurz die Installation der Multiboot und Backup-Infrastruktur auf eigene Rechner erklärt.

4.1 Einrichten des Servers

Die Multiboot und Backup-Infrasruktur benötigt einen Server im Netzwerk. Auf dem Server muss folgendes installiert werden:

1. Installieren Sie auf dem Server im Netzwerk einen DHCP Server. Konfigurieren Sie die Datei /etc/dhcpd.conf. Eine Beispieldatei finden Sie im Anhang (B). Neben grundlegenden Einstellungen wie IP-Adresse, Default-Gateway und Subnetmask, muss man Bootrom Informationen aber auch PXE-spezifische Informationen konfigurieren. Es empfiehlt sich die MAC Adresse der Netzwerkkarten der Clients an eine feste IP zu binden. Um PXE-spezifische Informationen zu konfigurieren, muss man ein neues option tag für die Vendor-Klasse hinzufügen. Der Name der neuen Option muss auf ClassID, der Typ auf String und der Identifikator auf 60 gesetzt werden. Die neue Option muss nun in das Rahmenprogramm eingefügt werden und der Wert auf PXEClient gesetzt werden. Vendor encapsulated options werden definiert durch Option 43. Möchte man keine PXE spezifischen Optionen setzen, so muss man diesen Wert auf folgendes Array aus Bytes setzen: 01,04,00,00,00,00,ff. Der Name der Batchfile muss angegeben werden, damit mit Hilfe des TFTP die richtige Batchfile beim Bootvorgang der Clients kopiert werden kann.

2. Installieren Sie anschließend auf dem Server einen TFTP Server. Der TFTP Server muss auf dem gleichen PC installiert werden wie zuvor der DHCP Server. Achten Sie hierbei auf auftretende Versionskonflikte. Nicht jeder TFTP Server arbeitet einwandfrei mit BPBatch zusammen. Bei tftp-hpa Version 0.28 treten keine Probleme auf.

3. Installieren Sie nun auf dem Server einen NFS Server. Das NFS System wird für das Backupsystem und für die erstmalige Konfiguration der Clients benötigt. Installieren Sie in das Verzeichnis /backup/restore das Restoresystem. Ergänzen Sie die Datei /etc/exports um den Eintrag /backup/restore 192.168.100.0/24(ro,no_root_squash) und passen Sie die IP in diesem Eintrag entsprechend an.

4. Erstellen Sie auf dem Server das Verzeichnis /tftpboot und kopieren Sie alle unten stehenden Dateien, die im Verlaufe des Boot-Vorgangs den Clients zur Verfügung gestellt werden sollen, in dieses Verzeichnis. Die Dateien befinden sich auf der beiliegenden CD. Der Verzeichnis-Name /tftpboot kann in der /etc/dhcpd.conf geändert werden.
   Es wird einmal BpBatch benötigt. Sie können BpBatch von www.bpbatch.org bekommen und

müssen lediglich die `bpbatch.bpb` (Beispieldatei im Anhang B) ändern. Das im Startmenü verwendete Hintergrundbild (z.B. `boot.gif`) und die verwendete Schriftart (z.B. `menu.fnt`) müssen Sie ebenfalls in diesem Verzeichnis zur Verfügung stellen. Das Hintergrundbild muss als 8Bit `Pcx` oder als `Gif` gespeichert sein, ansonsten kann es passieren, dass Bpbatch das Hintergrundbild nicht anzeigt. 24Bit Bilder zeigt BpBatch nicht an. Kopieren Sie das Windowsimage (z.B. `win98.imz`) und die Windowsdateien, die Sie bei jedem Startvorgang neu patchen möchten, in dieses Verzeichnis (z.B. `autoexec.ref` und `patch.ref`). Die `autoexec.ref` überschreibt die Datei `autoexec.bat` und `patch.ref` die Datei `/temp/patch.reg` und gibt dem Rechner einen dynamischen, von der IP abhängigen Rechnernamen, um zu vermeiden, dass mehrere Rechner mit dem selben Namen in einem Netzwerk auftreten, da dieses unter Windows zu einer Fehlermeldung beim Starten führt. Desweiteren müssen sich alle verwendeten Linux-Kernel und die Initial-Ramdisk in diesem Verzeichnis befinden (z.B. `suse.krn` und `rnplinux.krn` und `cinitrd.gz`). Alle Linuxdateien die Sie bei jedem Linuxstart neben dem Kernel und der Initrd noch neu kopieren, müssen ebenfalls in diesem Verzeichnis liegen (z.B. `boot.b` und `chain.b` und `boot.msg`). Für die Installation auf den Clients ist es noch wichtig in diesem Verzeichnis das verwendete Linux Defaultimage zur Verfügung zu stellen. Verwenden Sie die Beispieldatei (`bpbatch.bpb`), so passen Sie neben den oben verwendeten Dateinamen, die IP-Adressen des DNS, Gateways, NFS und die verwendete Hostmask (Zeile 61-69 in der Beispieldatei im Anhang B) an Ihr System an.

4.2 Installation der Clients

Die Installation der Clients ist relativ einfach. Tragen Sie auf dem Server, sofern noch nicht geschehen, die neuen Mac-Adressen der Netzwerkkarten der Clients in die `/etc/dhcpd.conf` ein und ordnen Sie ihnen jeweils eine IP-Adresse zu. Starten Sie nun auf allen Clients das Bootmenü und führen Sie den Punkt `IT-Sicherheit Restore` aus. Mit Hilfe einer Initial Ramdisk wird das auf dem NFS befindende Restoresystem gebootet. Partitionieren Sie nun die Festplatte der Clients mit Hilfe des ausgeführten Scriptes neu und kopieren Sie sowohl das Defaultimage für SECP Gruppe 1 als auch das Defaultimage für die SECP Gruppe 2 auf die Clients. Die Partitionierung kann vom Admin direkt in der Datei `config.pl` (siehe Anhang B config.pl Zeile 60) vorgenommen werden. Der Vorgang wird je nach Schnelligkeit des Netzwerkes einige Zeit in Anspruch nehmen. Nach der Installation des ersten Defaultimages, bootet der Rechner neu. Es muss wieder der Punkt `IT-Sicherheit Restore` ausgewählt werden, bevor mit der zweiten Gruppe fortgefahren werden kann. Als letztes starten Sie den Punkt `RNP Win Restore`, um das Windowssystem in die lokale Cache-Partition zu kopieren.

4.3 Erzeugen von Images

4.3.1 Windowsimage

Images werden mit Hilfe der bei BPBatch mitgelieferten Programme MrZip und MrBatch erstellt. Bei der Erstellung eines Windows Images ist zu beachten, das kein Image von einem Windows System, das gerade in Betrieb ist, erzeugt wird, da, durch die beschränkte Windowsfreigabe von sich in Betrieb befindenden Dateien, Fehler auftreten können. Genauso ist zu beachten, dass das System, unter dem ein Windows Image erzeugt wird, mit den langen Dateinamen, die unter Windows verwendet werden, umzugehen vermag, was ein Erstellen eines Windows Images unter DOS ausschließt. Zum Erstellen eines Windows‘98 Image, wie es für die Praktika benötigt wird, wird das Programm MrZip.exe unter Windows oder MrZip unter Linux gestartet. Eine mögliche Eingabe in dem Programm MrZip könnte folgendermaßen aussehen:

```
# Script für MRzip zum Erstellen eines Win98 OS image
showlog
#Filtern unnötiger Dateien
```

filter -“*.swp“
filter -“*.tmp“
filter -“*.pwl“
filter -“*.bak“
filter -“*.old“
filter -“*.reg“
filter -“*.cab“

Filtern unnötiger Verzeichnisse
filter -“windows/cookies/*.*“
filter -“temp/*“
filter -“windows/temp/*“
filter -“windows/history/*“
filter -“windows/Temporary Internet Files/*.*“
filter -“windows/verlauf/*.*“
filter -“windows/recent/*“
filter -“windows/Profiles/sysadmin/cookies/*.*“
filter -“windows/Profiles/sysadmin/his6/*.*“
filter -“windows/msimgsiz.dat“

Komprimiere Laufwerk C:
fullzip “c:/“ “c:/temp/win98.imz“

Um das erstellte Image mit BpBatch verwenden zu können, muss es in das Verzeichnis `/tftpboot` kopiert werden. Weicht der Name des Image von `win98.imz` ab, so ist der Name entsprechend auch in der BpBatch Konfigurationsfile `BpBatch.bpb` in Zeile 152 zu ändern.
Die Images sind nicht portabel und müssen für jeden Mandanten neu gebaut werden, sofern er nicht exakt die gleiche Hardware besitzt. Auch eine andere Reihenfolge der Karten im Rechner ist für Windows andere Hardware.

4.3.2 Linuximage

Im Gegensatz zum Windowsimage, ist das Erstellen eines Linuximages unproblematisch. Ein Image von einem Linuxsystem kann während dessen Betrieb mit dem Befehl `tar` erstellt werden. Die erstellte Imagefile muss BpBatch im Verzeichnis `/tftpboot` zur Verfügung gestellt werden. Ein Kommandozeilenbeispiel ist:

```
tar czf /tmpmnt/backup.tgz exclude /etc exclude /tmpmnt exclude /home
exclude /proc exclude /tmp /.
```

4.4 Update eines Betriebsystems

Entschließt man sich dazu, die verwendeten Betriebsysteme auf den Clients zu updaten, so muss man folgende Schritte durchführen:

4.4.1 Update von Suse 8.0 auf Suse 9.0

Installieren Sie das Defaultimage von Suse 8.0 auf einen PC und führen Sie die von Suse zur Verfügung gestellte Funktion `Suse Upgrade` aus. Ist Suse 9.0 auf dem PC installiert, so erstellen Sie wie oben beschrieben ein neues Defaultimage und kopieren Sie es in das `/tftpboot` Verzeichnis auf dem Server.

4.4.2 Umstellen auf andere Betriebsysteme

Möchte man von Suse Linux auf beispielsweise Mandrake Linux umstellen, so kann man das bisher verwendete Defaultimage nicht weiter verwenden. Es muss wie oben beschrieben ein neues Defaultimage angelegt werden und in das Verzeichnis `/tftpboot` auf den Server kopiert werden. Update eines Windowsimage geht analog wie oben beschrieben.

4.5 SECP Backup/Restore Befehle

Jeder Gruppe stehen fünf Speicherplätze (Slots) für Backups zur Verfügung. Die Backups werden immer inkrementell (level 1) zum Defaultimage erstellt. Im folgenden werden die mit dem Backup/Restore Vorgang zusammenhängenden Befehle dargestellt. Im Normalfall sollte ein Aufruf der Befehle ohne Parameter ausreichend sein.

4.5.1 listbackups

Aufruf: listbackups [SLOT]
Funktion: Listet alle bereits erstellten Backups oder deren Inhalt.
Parameter: SLOT Ist eine Ziffer zwischen 1 und 5, die eine Auswahl eines bestimmten Backups ermöglicht. Falls dieser Parameter angegeben ist, wird der Inhalt (also die Dateien) des in SLOT gesicherten Backups angezeigt. Ein Aufruf ohne Parameter listet die Belegung der Slots.

4.5.2 dobackup

Aufruf: dobackup [SLOT]
Funktion: Sichert die Daten auf dem Server
Parameter: SLOT ist eine Ziffer zwischen 1 und 5, die eine Auswahl eines bestimmten Backups ermöglicht. Die bereits vorhandenen Backups mit zugehörigen Slot Ziffern, erhält man über listbackups. Falls der Parameter angegeben ist, wird das Backup in dem gewählten Slot durch ein neues Backup ersetzt.
DefaultWert: nächster freier Slot oder, falls alle Slots belegt sind, ältester Slot.

4.5.3 dorestore

Aufruf: dorestore [EXTMASK [SLOT]]
Funktion: Stellt Backups wieder her.
Parameter: EXTMASK spezifiziert die zu extrahierenden Dateien.
Default Wert: /
SLOT ist eine Ziffer zwischen 1 und 5, die eine Auswahl eines bestimmten Backups ermöglicht. Die bereits vorhandenen Backups mit zugehörigen Slot Ziffern, erh¨alt man über listbackups.
Default Wert: letztes gesichertes Backup.

4.5.4 Beispiele

dorestore /etc/HOSTNAME 3
stellt die Datei /etc/HOSTNAME aus dem Backup in Slot 3 wieder her
dorestore /etc/HOSTNAME
stellt die Datei /etc/HOSTNAME aus dem letzen gesicherten Backup wieder her
dorestore

stellt alle Dateien aus dem letzen gesicherten Backup wieder her

4.5.5 Anmerkungen

Da die Befehle zentral auf dem Server verarbeitet werden und die Verbindung durch ssh verschlüsselt ist, benötigt man ein Passwort. Dieses Passwort ist gleich dem default root- Passwort auf den IT-Sicherheits Rechnern. Beim ersten Aufruf eines dieser Befehle kann es sein, daß eine Anfrage kommt ob man den ssh-Hostkey des Servers cachen möchte. Diese Frage einfach mit yes beantworten.

4.6 Kryptographisches Filesystem

Auf allen Rechnern befindet sich pro Gruppe ein verschlüsseltes Filesystem. Die root Partition soll nur in Abhängigkeit des richtigen Partitionspasswortes erfolgreich gemounted werden. Standardmäßig ist das Passwort des Kryptographischen Filesystems gleich dem root Passwort. Daher sollte sofort nach dem ersten booten des Systems das Passwort sowohl für den root Account, als auch für das Filesystem unverzüglich geändert werden. Für das Ändern des Passwortes steht dem Benutzer der Befehl `chcryptopw` (siehe 3.1.2) zur Verfügung. Zuerst wird der Benutzer aufgefordert sein altes Passwort einzugeben und danach kann er durch zweimalige Eingabe des gewünschten neuen Passwortes die Änderung vollziehen.

4.7 Änderung der Partitionierung

Die Partitionierung kann vom Admin direkt in der Datei `config.pl` (siehe Anhang B config.pl Zeile 60) vorgenommen werden. Die Eigenschaften jeder Partition werden durch vier Parameter bestimmt: Größe der Partition, Typ der Partition, Bootfähigkeit und durch mkfs Optionen. Krypto-Partitionen können nur indirekt über die `config.pl` erzeugt und geändert werden (siehe B config.pl Zeile 83). Das heißt, um eine Krypto-Partition zu erzeugen oder zu ändern, wird ein Defaultimage erzeugt, die Partitionierung in der `config.pl` geändert, und das Defaultimage wieder auf den PC kopiert.

Kapitel 5

Zusammenfassung

Die in diesem Systementwicklungspraktikum entworfene und realisierte Infrastruktur bietet den Teilnehmern der Praktika ein komfortables über die Cursertasten einfach zu bedienendes graphisches Bootmenü, welches mit BPBatch umgesetzt wurde. Die Infrastuktur ist voll mandantenfähig und bietet den Benutzern Sicherheit durch Abschotten der Mandanten mit Hilfe eines Kryptofilesystems. Zusätzlich kann ein Benutzer der Infrastruktur selbständig Backups von seinen Daten machen, die zentral gespeichert werden. Um die Anforderungen der Sicherheit zu erfüllen, wird auch der Datentransfer beim Übertragen der Backups verschlüsselt.

Anhang A

Quelltexte

Listing A.1: Initscript

```
#! /bin/sh

export PATH=/bin:/sbin

CRYPTOPART="/dev/hda6"
SECPGROUP="itsec1"
MTRUE=0
loadkeys /lib/de.map
insmod loop_fish2
mount /dev/hda2 /tmpmnt -t vfat
echo "Setting up: local bootmanager..."
lilo
echo "Mounting crypto-root-filesystem for $SECPGROUP..."

cd /tmpmnt/$SECPGROUP

while test "$MTRUE" -ne "1"; do
  MTRUE=1
  echo "Enter your password:"
  stty -echo
  read PW
  KEY=`sh -c "echo $PW" | gpg --batch --passphrase-fd 0 --no-options -d user.key
       2> /dev/null`

  if test -z "$KEY"; then
    KEY=`sh -c "echo $PW" | gpg --batch --passphrase-fd 0 --no-options -d master.key
         2> dev/null`
  fi
  echo $KEY | losetup -e twofish -p 0 /dev/loop1 $CRYPTOPART 2> /dev/null

  stty echo
  mount -r -t ext3 /dev/loop1 /mnt || {
     echo "Password is wrong."
     losetup -d /dev/loop1 2> /dev/null
     MTRUE=0
  }

done
cd /mnt
umount /tmpmnt
pivot_root . old-root 2> /dev/null
exec chroot . sh -c 'exec /sbin/init' <dev/console >dev/console 2>&1
```

Listing A.2: chcryptopw.sh

```
#!/bin/sh
SECPGROUP=`cat /etc/SECPGROUP`
umount /mnt 2> /dev/null
mount /dev/hda2 /mnt -t vfat 2> /dev/null
cd /mnt/$SECPGROUP
```

```
6  echo "Changing passphrase of partition key..."
7  echo "Enter your current passphrase and then the new passphrase twice..."
8  KEY=`gpg --no-options --decrypt user.key`
9  echo $KEY | gpg --no-options --cipher-algo cast5 -c -o /tmp/new.key
10 ERR=`echo dummy | gpg --no-options --passphrase-fd 0 --batch -vv /tmp/new.key 2>&1
       | grep symkey`
11 if ! test -z "$ERR" ; then
12   mv -f /tmp/new.key user.key
13   echo "Success. Passphrase changed."
14 else
15   echo "Failed. Passphrase not changed."
16 fi
17 umount /mnt 2> /dev/null
```

Listing A.3: dobackup.sh

```
1 #!/bin/sh
2 SERVER=`cat /etc/SECPSERVER`
3 GROUP=`cat /etc/SECPGROUP`
4 PASSWD=`cat /etc/SECPPASSWD`
5 SLOT=$1
6 if test -z $SLOT ; then
7   SLOT="default"
8 fi
9 ssh -l userback $SERVER "backup $GROUP $PASSWD $SLOT"
```

Listing A.4: dorestore.sh

```
1  #!/bin/sh
2  SERVER=`cat /etc/SECPSERVER`
3  GROUP=`cat /etc/SECPGROUP`
4  PASSWD=`cat /etc/SECPPASSWD`
5  FILEMASK=$1
6  SLOT=$2
7  if test -z $SLOT ; then
8    SLOT="default"
9  fi
10 if test -z $FILEMASK ; then
11   FILEMASK="/"
12 fi
13 ssh -l userback $SERVER "restore $GROUP $PASSWD $SLOT:$FILEMASK"
```

Listing A.5: listbackups.sh

```
1 #!/bin/sh
2 SERVER=`cat /etc/SECPSERVER`
3 GROUP=`cat /etc/SECPGROUP`
4 PASSWD=`cat /etc/SECPPASSWD`
5 SLOT=$1
6 if test -z $SLOT ; then
7   SLOT="default"
8 fi
9 ssh -l userback $SERVER "list $GROUP $PASSWD $SLOT"
```

Listing A.6: Master Notfall Backup/Restore Script

```
1  #!/usr/bin/perl
2
3  $| = 1; # flush always the output buffer
4
5  require "config.pl";
6
7  ########
8  # Subs #
9  ########
10
11 # forward definitions
12 # do some initialisation stuff
13 sub Init;
14
15
16 # returns the MAC to a given IP
```

```
17 sub getMacFromIp;
18 # returns the pcname to a given MAC
19 sub getNameFromMac;
20 # get Partition Infos
21 sub getPartition;
22 # check if Partition is valid
23 sub validPartition;
24 # setup the Partitions
25 sub setupPartitions;
26 # mkfs on Partition
27 sub mkfsPartition;
28
29 # Backup subs
30 sub doBackup;
31
32 # Restore subs
33 sub doRestore;
34
35 # menus for Backup/Restore
36 sub menu_backuprestore;
37 sub menu_choosepc;
38 sub menu_choosegroup;
39 sub menu_chooseslot;
40 sub menu_confirmselection;
41
42 sub doExec {
43   my $importance = shift;
44   my $args = join " ", @_;
45   print "[DEBUG] doing $sshpath $args\n";
46   print `$sshpath $args`;
47 }
48
49 ########
50 # Main #
51 ########
52
53
54 Init();
55 menu_backuprestore();
56 print "\n"x2;
57 menu_choosepc();
58 print "\n"x2;
59 menu_choosegroup();
60 print "\n"x2;
61 menu_chooseslot();
62 print "\n"x2;
63 menu_confirmselection();
64
65 if ($u_action eq "backup") {
66   doBackup();
67 } else {
68   doRestore();
69 }
70
71
72 # RESTORE
73 # ip <=> mac => suggested comp
74 # choose pc
75 # choose group
76 # choose image slot
77 # do restore
78 # BACKUP
79 # ip <=> mac => suggested comp
80 # choose pc
81 # choose group
82 # choose image slot
83 # do backup
84
85
86 sub printCaption {
87   my $caption = shift;
88   print "\n$caption\n" . "=" x length($caption) . "\n\n";
89 }
90
91 sub Init {
```

```
92    my $tmpvar;
93
94    foreach $key (keys %macs) {
95      $names{$macs{$key}} = $key;
96    }
97  # ssh env todo
98    $tmpvar = $ENV{"SSH_CLIENT"};
99    if (($tmpvar =~ /(\d+)\.(\d+)\.(\d+)\.(\d+)\s+\d+\s+\d+$/)) {
100     $u_ip = "$1.$2.$3.$4";
101   } else {
102     die "Can't find env var SSH_CLIENT.\n";
103   }
104   $sshpath .= " $u_ip";
105   printCaption("ITSEC Backup/Restore System (your ip: $u_ip)");
106   doExec(0, "'$ntpdatepath -b $timeserver ; $hwclockpath --systohc'");
107 }
108
109 sub doBackup {
110   $groupscript = $scriptdir. "/" .$u_group. "_" . $u_action;
111   print "Spawning backup script $groupscript...\n";
112   system("$groupscript", "$u_pc", "$u_ip", "$u_slot", "$u_group");
113 }
114
115 sub doRestore {
116   restorePartitions();
117   $groupscript = $scriptdir. "/" .$u_group. "_" . $u_action;
118   print "Spawning restore script $groupscript...\n";
119   system("$groupscript", "$u_pc", "$u_ip", "$u_slot", "$u_group");
120 }
121
122
123 sub getMacFromIp {
124   my $ip = shift;
125   if (`$arppath -an $ip` =~ /\($ip\)\sat\s(\w{2}:\w{2}\:\w{2}\:\w{2}\:\w{2}\:\w{2})\s
        /)  {
126     return $1;
127   }
128 }
129
130 sub getNameFromMac {
131   my $mac = shift;
132   return $macs{$mac};
133 }
134
135
136 sub menu_backuprestore {
137   my $mychoice=-7;
138   printCaption("What do you want to do?");
139   print "[1] Restore\n";
140   print "[5] Backup\n";
141   while ($mychoice == -7) {
142     print "\nEnter your choice: ";
143     $input = <STDIN>;
144     chomp $input;
145     SWITCH: {
146       ($input == 1) and ($mychoice = 1), last SWITCH;
147       ($input == 5) and ($mychoice = 2), last SWITCH;
148     }
149   }
150   $u_action = ($mychoice == 1 ? "restore" : "backup" );
151 }
152
153 sub menu_choosepc {
154   my $input, $your_mac;
155   my $suggestion;
156
157   printCaption("Choose your Hostname");
158
159   $your_mac = getMacFromIp($u_ip);
160   $suggestion = getNameFromMac($your_mac);
161   print "Valid names: " . (join ', ', sort keys %names);
162   $input = 'kein scherz';
163   while (not exists $names{$input}) {
164     print "\nEnter a valid PC Accountname [$suggestion]: ";
165     $input = <STDIN>;
```

```
166     chomp $input;
167     $input = $suggestion if ($input eq "");
168   }
169   $u_pc = $input;
170 }
171
172 sub menu_choosegroup {
173   my $input=-7;
174   my @mygroups = sort keys %groups;
175   printCaption("Choose your Group");
176   for ($i = 0; $i <= $#mygroups; $i++) {
177     print "[".($i+1)."] ".$mygroups[$i]."\n";
178   }
179   while ($input < 1) {
180     print "\nEnter your choice: ";
181     $input = <STDIN>;
182     chomp $input;
183   }
184   $u_group = $mygroups[$input-1];
185 }
186
187 sub menu_chooseslot {
188   my @usedslots;
189   my @options;
190   my $input;
191   my $selected = "";
192   my $numused = 0;
193   my $sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst;
194
195   printCaption("Choose your Slot");
196
197   opendir SLOTDIR, "$pooldir/$u_pc/$u_group/" or die "Fatal: Directory $pooldir/$u_pc
        /$u_group does not exist!!\n";
198   @usedslots = sort (grep /.gz$/,(grep !/^.\.?$/, readdir SLOTDIR));
199   closedir SLOTDIR;
200
201   print "Used Slots:\n";
202   $numused = $#usedslots + 1;
203   if ($numused > 0) {
204     for ($i = 0; $i < $numused; $i++) {
205       $usedslots[$i] =~ /(\d+).gz/;
206       print "[".($i+1)."] ".$usedslots[$i]."\tDate:\t". scalar localtime($1) ."\n";
207     }
208   }
209   print "[*] No slots used\n" if ($numused == 0);
210   print "\n[8] default slot\n\n";
211   print "[10] default image only\n";
212   print "Free slots: ". ($num_slots - $numused) . "/$num_slots\n";
213
214   while ($selected eq "") {
215     print "\nEnter your choice[default]: ";
216     $input = <STDIN>;
217     chomp $input;
218     SWITCH: {
219       ($input == 10) and ($selected = "defaultonly"), last SWITCH;
220       (($input > 0) && ($input <= $numused)) and ($selected = $usedslots[$input-1]),
            last SWITCH;
221       ($input == 8) and ($selected = "default"), last SWITCH;
222       ($input eq "") and ($selected = "default"), last SWITCH;
223     }
224   }
225 #default behaviour for selecting an entry
226 #for restore select the newest one
227 #for backup the next free or the oldest one
228   if ($selected eq "default") {
229
230     if ($u_action eq "restore") {
231       $selected = "defaultonly";
232       $selected = $usedslots[$numused-1] if ($numused > 0);
233     } else {
234       $selected = time . ".gz" if ($numused < $num_slots);
235       $selected = $usedslots[0] if ($numused >= $num_slots);
236     }
237   }
238   $u_slot = $selected;
```

```
239 }
240
241 sub menu_confirmselection {
242   printCaption("Do you want to continue ?");
243
244   print "*"x30 . "\n";
245   print "Your IP: $u_ip\n";
246   print "Action: $u_action\n";
247   print "Hostname: $u_pc\n";
248   print "Group: $u_group\n";
249   print "Image/Slot: $u_slot\n";
250   print "*"x30 . "\n"x2;
251   print "If these settings are correct, feel free to enter 'yes'\n";
252   while ($input ne "yes") {
253     print "\nEnter yes or no to proceed: ";
254     $input = <STDIN>;
255     chomp $input;
256     die("$u_action aborted by User\n") if ($input eq "no");
257   }
258 }
259
260 sub getPartition {
261   my $partid = shift;
262   my $mypart = quotemeta($myhd . $partid);
263   foreach ( split "\n", `$sshpath $sfdiskpath -lx -uM -n $myhd`) {
264     if (/^$mypart\s+\W?\s+(\d+)\W?\s+(\d+)\W?\s+(\d+)\W?\s+(\d+)\W?\s+(\w+)/) {
265         # start end size blocks type
266         return ($1, $2, $3, $4, $5);
267     }
268   }
269   return 0;
270 }
271
272 sub validPartition {
273   my $partid = shift;
274   my $start, $end, $size, $blocks, $type;
275   ($start, $end, $size, $blocks, $type) = getPartition($partid);
276   if ((($partitions{"$partid"}[0] <= $size) && ($partitions{"$partid"}[1] == $type)))
          {
277     return 1;
278   }
279   return 0;
280 }
281
282 sub setupPartitions {
283   my $start,$end,$blks,$size,$type,$boot;
284   my $myparttable = "";
285   open(SFDISK, "|$sshpath '$sfdiskpath -uM -D $myhd'");
286   $start = 0;
287   foreach $part (sort keys %partitions) {
288     $size = $partitions{"$part"}[0];
289     $type = $partitions{"$part"}[1];
290     $boot = $partitions{"$part"}[2];
291
292     $myparttable .= "$start,$size,$type,$boot\n";
293     $start = "";
294   }
295   print SFDISK $myparttable;
296   close(SFDISK);
297 }
298
299 sub mkfsPartition {
300 # formatieren
301   my $part = shift;
302   my $start, $end, $size, $blocks, $type, $options;
303   ($start, $end, $size, $blocks, $type) = getPartition($part);
304   $options = $partitions{"$part"}[3];
305   $mymkfs = $mkfs{"$type"};
306   if ($mymkfs) {
307     print "Doing $mymkfs $options on $myhd$part ...\n";
308     print `$sshpath '$mymkfs $options $myhd$part'`;
309   }
310   return 1;
311 }
312
```

```
313 sub restorePartitions {
314   my $part;
315   my $dorestore = 0;
316   foreach $part (sort keys %partitions)  {
317     if (validPartition($part) == 0) {
318       $dorestore = 1;
319       print "Partition $part not valid\n";
320     }
321   }
322   if ($dorestore) {
323     setupPartitions();
324     #formatiere nur wenn Partition für uns interessant ist ...
325     foreach $part (sort keys %partitions) {
326       if (join(' ', @{$groups{$u_group}}) =~ /$part/) {
327         mkfsPartition($part);
328       }
329     }
330   } else {
331     print "All Partitions valid. Skipping fdisk.\n";
332     mkfsPartition($install{"$u_group"}[0]);
333     mkfsPartition($install{"$u_group"}[1]);
334   }
335 }
```

Listing A.7: User Backup/Restore Script

```
1  #!/usr/bin/perl
2
3  $| = 1; # flush always the output buffer
4
5  require "config.pl";
6
7  ########
8  # Subs #
9  ########
10
11 # forward definitions
12 # do some initialisation stuff
13 sub Init;
14
15
16 # returns the MAC to a given IP
17 sub getMacFromIp;
18 # returns the pcname to a given MAC
19 sub getNameFromMac;
20
21 # Backup subs
22 sub doBackup;
23
24 # Restore subs
25 sub doRestore;
26
27 # other subs
28 sub choosepc;
29
30 ########
31 # Main #
32 ########
33
34 Init();
35 choosepc();
36 print "\n"x2;
37 validate_user();
38 if ($u_action eq "backup") {
39   doBackup();
40 }
41 if ($u_action eq "restore") {
42   doRestore();
43 }
44 if ($u_action eq "list") {
45   doList();
46 }
47
48 sub doExec {
49   my $importance = shift;
```

```
50   my $args = join " ", @_;
51 #  print "[DEBUG] Doing $u_sshpath $args\n";
52   print `$u_sshpath $args`;
53 }
54
55 sub printCaption {
56   my $caption = shift;
57   print "\n$caption\n" . "=" x length($caption) . "\n\n";
58 }
59
60 sub Init {
61   my $tmpvar;
62
63   die("Not enough parameters given") if ($#ARGV != 1);
64   ($u_action, $u_group, $u_pass, $u_slot) = split ' ', $ARGV[1];
65
66   foreach $key (keys %macs) {
67     $names{$macs{$key}} = $key;
68   }
69   $tmpvar = $ENV{"SSH_CLIENT"};
70   if (($tmpvar =~ /(\d+)\.(\d+)\.(\d+)\.(\d+)\s+\d+\s+\d+$/)) {
71     $u_ip = "$1.$2.$3.$4";
72   } else {
73     die "Can't find env var SSH_CLIENT.\n";
74   }
75   $u_sshpath .= " $u_ip";
76   printCaption("ITSEC Backup/Restore System (ip: $u_ip, group: $u_group)");
77   doExec(0, "'$ntpdatepath -b $secpserver ; $hwclockpath --systohc'");
78 }
79
80 sub doBackup {
81   my @usedslots, $numused, $newimage,$timestamp,$selected;
82
83   opendir SLOTDIR, "$pooldir/$u_pc/$u_group/" or die "Fatal: Directory $pooldir/$u_pc
         /$u_group does not exist!!\n";
84   @usedslots = sort (grep /.gz$/,(grep !/^.\.?$/, readdir SLOTDIR));
85   closedir SLOTDIR;
86   $numused = $#usedslots + 1;
87   $selected = $u_slot;
88   if (($selected > 0) && ($selected <= $numused)) {
89     unlink "$pooldir/$u_pc/$u_group/".$usedslots[$selected-1];
90   } else {
91     unlink "$pooldir/$u_pc/$u_group/".$usedslots[0] if ($numused >= $num_slots);
92   }
93
94   $u_slot = time . ".gz";
95   open(TSFILE, "$pooldir/$u_pc/$u_group/restore.time");
96   $timestamp = <TSFILE>;
97   chomp $timestamp;
98   $timestamp = scalar localtime($timestamp);
99   close(TSFILE);
100  $newimage = "$pooldir/$u_pc/$u_group/$u_slot";
101  print "Destination: $u_slot\n";
102  $rootpart = "/";
103  #print "Remounting $rootpart readonly...\n";
104  #doExec(1, "'/bin/mount -n -o remount,ro $rootpart'");
105  print "Doing the backup...\n";
106  print `$u_sshpath "$dumppath -1 -T '$timestamp' -f - $rootpart" | $gzippath -c >
         $newimage`;
107  #print "Remounting $rootpart read-write...\n";
108  #doExec(1, "'/bin/mount -n -w -o remount $rootpart'");
109 }
110
111 sub doRestore {
112   my $newimage, $filemask;
113   ($u_slot, $filemask) = split ':', $u_slot;
114   if (($filemask eq "") || ($u_slot eq "")) {
115     die("Wrong parameters.\n");
116   }
117   chooseslot();
118   $newimage = "$pooldir/$u_pc/$u_group/$u_slot";
119   print "Restoring ...\n";
120   print `$gzippath -cd $newimage | $u_sshpath 'cd / ; $restorepath -xf - $filemask
         2> /dev/null'`;
121   print "Done!\n";
```

```
122 }
123
124
125 sub getMacFromIp {
126   my $ip = shift;
127   if (`$arppath -an $ip` =~ /\($ip\)\sat\s(\w{2}:\w{2}\:\w{2}\:\w{2}\:\w{2}\:\w{2})\s
        /)  {
128     return $1;
129   }
130 }
131
132 sub getNameFromMac {
133   my $mac = shift;
134   return $macs{$mac};
135 }
136
137
138 sub validate_user {
139   die("unknown command.") if (($u_action ne "backup") && ($u_action ne "restore")
         && ($u_action ne "list"));
140   die("unknown group.") if (not exists $groups{"$u_group"});
141   if ($secpauth{"$u_pc-$u_group"} eq "") {
142     die("There is no authline for your account\n");
143   }
144   die("group/password combination is wrong.") if (($u_pass ne $secpauth{"$u_pc-
        $u_group"}));
145 }
146
147 sub choosepc {
148   my $your_mac;
149   my $suggestion;
150   $your_mac = getMacFromIp($u_ip);
151   $suggestion = getNameFromMac($your_mac);
152   if ($suggestion eq "") {
153 # comment the following line for very strict access settings
154     die("Can't get your MAC, Access denied.");
155     print "Can't get your MAC... trying auth with hostname.\n";
156     $suggestion = `$u_sshpath 'cat /etc/HOSTNAME'`;
157     if (($suggestion =~ /^(\w+)\./) || ($suggestion =~ /^(\w+)$/ )) {
158       $suggestion = $1;
159     }
160     if (not exists $names{$suggestion})  {
161       die("Can't identify your Host.");
162     }
163   }
164   $u_pc = "$suggestion";
165   print "Identified your Host as $u_pc\n";
166 }
167
168 sub doList {
169   my @usedslots;
170   my $numused = 0;
171   my $selected;
172   my $newimage;
173
174   opendir SLOTDIR, "$pooldir/$u_pc/$u_group/" or die "Fatal: Directory $pooldir/$u_pc
        /$u_group does not exist!!\n";
175   @usedslots = sort (grep /.gz$/,(grep !/^.\.?$/, readdir SLOTDIR));
176   closedir SLOTDIR;
177   $numused = $#usedslots + 1;
178
179   $selected = $u_slot;
180   if (($selected > 0) && ($selected <= $numused)) {
181     printCaption("Content of slot $selected:");
182     $u_slot = $usedslots[$selected-1];
183     $newimage = "$pooldir/$u_pc/$u_group/$u_slot";
184     print `$gzippath -cd $newimage | $u_sshpath 'cd / ; $restorepath -tf - 2> /dev/
          null'`;
185   } else {
186   printCaption("Available slots:");
187   if ($numused > 0) {
188     for ($i = 0; $i < $numused; $i++) {
189       $usedslots[$i] =~ /(\d+).gz/;
190       print "[".($i+1)."] ".$usedslots[$i]."\tDate:\t". scalar localtime($1) ."\n";
191     }
```

```
192   } else {
193     print "none\n\n";
194   }
195   }
196 }
197
198 sub chooseslot {
199   my @usedslots;
200   my @options;
201   my $input;
202   my $selected = "";
203   my $numused = 0;
204   my $sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst;
205
206   opendir SLOTDIR, "$pooldir/$u_pc/$u_group/" or die "Fatal: Directory $pooldir/$u_pc
          /$u_group does not exist!!\n";
207   @usedslots = sort (grep /.gz$/,(grep !/^.\.?$/, readdir SLOTDIR));
208   closedir SLOTDIR;
209
210   $numused = $#usedslots + 1;
211   die("No slots available.") if ($numused < 1);
212 # select last backup by default
213   $selected = $u_slot;
214   $selected = $numused if ($u_slot eq "default");
215   die("Slot does not exist.") if (!(($selected > 0) && ($selected <= $numused)));
216   $u_slot = $usedslots[$selected-1];
217   $usedslots[$selected-1] =~ /(\d+).gz/;
218   print "Slot $selected (".scalar localtime($1)." selected. \n";
219 }
```

Listing A.8: ITSec Backup Script

```
1  #!/usr/bin/perl
2
3  require "config.pl";
4
5  my $rootpart;
6  my $bootpart;
7
8  sub Init;
9  sub cryptoSetup;
10 sub doMount;
11 sub createImage;
12 sub doUnmount;
13 sub cryptoTeardown;
14
15 Init();
16 cryptoSetup();
17 createImage();
18 cryptoTeardown();
19
20 sub Init {
21   if ($#ARGV != 3) {
22     die "Incorrect parameter count. Quitting!\n";
23   }
24   $u_pc = $ARGV[0];
25   $u_ip = $ARGV[1];
26   $u_slot = $ARGV[2];
27   $u_group = $ARGV[3];
28   $u_action = "backup";
29   $sshpath .= " $u_ip";
30
31   $rootpart = "/dev/loop1";
32   print "Starting backup for Host $u_pc($u_ip) using slot $u_slot...\n\n";
33 }
34
35 sub doExec {
36   my $importance = shift;
37   my $args = join " ", @_;
38   print "[DEBUG] Doing $sshpath $args\n";
39   print `$sshpath $args`;
40 }
41
42
43 sub createImage {
```

```
  my $defaultimage,$newimage,$timestamp;
# create default image
  if ($u_slot eq "defaultonly") {
   $defaultimage = $customfiles."/".$install{"$u_group"}[2];
   print "Creating new default Image...\n";
#dump.static 0anf - /dev/loop5 | gzip -c > /tmp/outfile.dump.gz
    print `$sshpath '$dumppath 0anf - $rootpart' | $gzippath -c > $defaultimage`;
    open(TSFILE, ">" . $customfiles."/".$install{"$u_group"}[2].".time");
    print TSFILE time;
    close(TSFILE);
  } else {
    $newimage = "$pooldir/$u_pc/$u_group/$u_slot";
# timestamp for incremental backup ....
#    open(TSFILE, $customfiles."/".$install{"$u_group"}[2].".time");
    open(TSFILE, "$pooldir/$u_pc/$u_group/restore.time");
    $timestamp = <TSFILE>;
    chomp $timestamp;
    $timestamp = scalar localtime($timestamp);
    close(TSFILE);
    print `$sshpath '$dumppath -1 -T "$timestamp" -f - $rootpart' | $gzippath -c >
        $newimage`;
  }
}

sub getcryptoKey {
  my $thefile = shift;
  return `$sshpath 'echo $master_pw | $gpgpath --batch --passphrase-fd 0 --decrypt
      $thefile '`;
}

sub cryptoSetup {
  my $part = $install{"$u_group"}[0];
  my $buf;
  my $partid = 2;
  my $key;
# extracting key ... and get pw ...
  print "Mounting unencrypted partition for crypto-setup... \n";
  doExec(0, "$mountpath $myhd$partid $instmptmp");
  $key = getcryptoKey("$instmptmp/$u_group/master.key");
  doExec(0, "$unmountpath $instmptmp");
#
##### expect script start ####
$buf = `$expectpath <<Expect_EOF
set timeout -1
spawn $sshpath $losetuppath -e twofish $rootpart $myhd$part
expect -exact "Password: "
send -- "$key\r"
expect eof
Expect_EOF`;
    print "$buf\n";
##### expect script end ####
  print "Mounting the / Partition ...\n";
  doExec(0, "$mountpath $rootpart $instmp");
}

sub cryptoTeardown {
  doExec(0, "$unmountpath $instmp");
  doExec(0, "'$losetuppath -d $rootpart'");
}
```

Listing A.9: ITSec Restore Script

```
#!/usr/bin/perl

$| = 1;

require "config.pl";

my $rootpart;
my $bootpart;

sub Init;
sub cryptoSetup;
sub doMount;
sub preInstall;
```

```
14 sub installImage;
15 sub postInstall;
16 sub doUnmount;
17 sub cryptoTeardown;
18
19 Init();
20 cryptoSetup();
21 doMount();
22 preInstall();
23 installImage();
24 postInstall();
25 doUnmount();
26 cryptoTeardown();
27
28 sub doExec {
29   my $importance = shift;
30   my $args = join " ", @_;
31   print "[DEBUG] doing $sshpath $args\n";
32   print `$sshpath $args`;
33 }
34
35 sub Init {
36   if ($#ARGV != 3) {
37     die "Incorrect parameter count. Quitting!\n";
38   }
39   $u_pc = $ARGV[0];
40   $u_ip = $ARGV[1];
41   $u_slot = $ARGV[2];
42   $u_group = $ARGV[3];
43   $u_action = "restore";
44   $sshpath .= " $u_ip";
45   $rootpart = "/dev/loop1";
46   print "Starting restore for Host $u_pc($u_ip) using slot $u_slot...\n\n";
47 }
48
49 sub setupKey {
50   my $key = shift;
51   my $pw = shift;
52   my $algo = shift;
53   my $keyfile = shift;
54   my $tmpfile = "$tmpdir/" . time . int(rand 20);
55   my $tmpvar;
56   $tmpvar = 'root@' . $u_ip;
57   open(TMPFILE, ">$tmpfile");
58   print TMPFILE $key;
59   close(TMPFILE);
60   print "Creating $keyfile...\n";
61   print `echo $pw | $gpgpath --batch --passphrase-fd 0 --cipher-algo $algo -c
       $tmpfile`;
62   print "Installing $keyfile on the client\n";
63   print `$scppath $tmpfile.gpg $tmpvar:$keyfile`;
64   unlink "$tmpfile", "$tmpfile.gpg";
65 }
66
67 sub cryptoSetup {
68     my $buf;
69     my $key;
70     my $part = $install{"$u_group"}[0];
71     my $partid = 2;
72     print "Mounting unencrypted partition for crypto-setup... \n";
73     doExec(0, "$mountpath $myhd$partid $instmptmp");
74     print "Creating directory $u_group on crypto-setup partition... \n";
75     doExec(0, "$mkdirpath $instmptmp/$u_group");
76
77     print "Generating crypto keyfiles...\n";
78     $key = `$genkey`;
79     chomp($key);
80     setupKey("$key", "$master_pw", "blowfish", "$instmptmp/$u_group/master.key");
81     setupKey("$key", "$user_pw", "cast5", "$instmptmp/$u_group/user.key");
82
83     print "Creating twofish encrypted root...\n";
84     # root partition
85 ##### expect script start ####
86 $buf = `$expectpath <<Expect_EOF
87 set timeout -1
```

```
spawn $sshpath $losetuppath -e twofish $rootpart $myhd$part
expect -exact "Password: "
send -- "$key\r"
expect eof
Expect_EOF`;
    print "$buf\n";
##### expect script end ####
    print "Creating ext3 filesystem on encrypted root...\n";
    doExec(1, "'$mkext2 -j $rootpart'");
    $part = $install{"$u_group"}[1];
    print "Setting up swap...\n";
    doExec(1, "'$mkswap $myhd$part'");
}

sub doMount {
  print "Mounting the / Partition ...\n";
  doExec(0, "$mountpath $rootpart $instmp");
}

sub preInstall {
}

sub installImage {
  my $defaultimage, $newimage;
  $defaultimage = $customfiles."/".$install{"$u_group"}[2];

# extract default image
  print "Restoring from Image - please wait...\n";
  #print `$catpath $defaultimage | $sshpath '$tarpath -pxzC $instmp' `;
  print "$gzippath -cd $defaultimage | $sshpath 'cd $instmp ; $restorepath -rf -'\n";
  print `$gzippath -cd $defaultimage | $sshpath 'cd $instmp ; $restorepath -rf -'`;
  if ($u_slot ne "defaultonly") {
    $newimage = "$pooldir/$u_pc/$u_group/$u_slot";
    print "Applying additional level 1 dump ...\n";
    print `$gzippath -cd $newimage | $sshpath 'cd $instmp ; $restorepath -f -'`;
  }
  print "Done!\n\n";
  open(TSFILE, ">$pooldir/$u_pc/$u_group/restore.time");
  print TSFILE time;
  close(TSFILE);
# extract selected diff todo

}

sub postInstall {
  my $myroot, $myswap, $tmpvar;
# host specific configuration
# clean proc, set hostname, set fstab, install kernel
  print "Cleaning /proc ...\n";
  doExec(0, "$rmpath -rf $instmp/proc/*");
  print "Setting up HOSTNAME ...\n";
  doExec(0, "'echo $u_pc > $instmp/etc/HOSTNAME'");
  doExec(0, "'echo $u_group > $instmp/etc/SECPGROUP'");
  $u_pass = $secpauth{"$u_pc-$u_group"};
  doExec(0, "'echo $u_pass > $instmp/etc/SECPPASSWD'");
  doExec(0, "'echo $secpserver > $instmp/etc/SECPSERVER'");


  $tmpvar = 'root@' . $u_ip;
  print "Copying custom files $customfiles/$u_group/* to $tmpvar:$instmptmp/$u_group
      /...\n";
  print `$scppath -pr $customfiles/$u_group/* $tmpvar:$instmptmp/$u_group/`;

  doExec(0, "'mkdir $instmp/root/.ssh'");
  print `$scppath $pubkeydir/* $tmpvar:$instmp/root/.ssh/`;
  print `$scppath -r $customfiles/bin/* $tmpvar:$instmp/sbin/`;

  $myroot = $rootpart . "\t/       ext3    defaults 1 2\n";
  $mysw = $myhd.$install{"$u_group"}[1] . "\tswap    swap    pri=42 0 0\n";

  print "Creating $instmp/etc/fstab...\n";
  open(FSTAB, "|$sshpath '$catpath > $instmp/etc/fstab'");
  print FSTAB $myroot . $mysw . $defaultfstab;
  close(FSTAB);
}
```

```
162
163 sub doUnmount {
164   #unmount
165   doExec(0, "'$unmountpath $instmp'");
166
167   # unmount
168   print "Unmounting partition for crypto-setup... \n";
169   doExec(0, "$unmountpath $instmptmp");
170 }
171
172 sub cryptoTeardown {
173   doExec(0, "'$losetuppath -d $rootpart'");
174 }
```

Anhang B

Konfigurationsdateien

Listing B.1: BPBatch Konfigurationsdatei

```
:startmenu
InitGraph "800x600"
Set CacheAlways="off"
Set CacheNever="on"
Set MenuNum=9
Set X=000
Set Y=400
Set MenuItem="RNP_Linux RNP_Windows SECP_Group_1 SECP_Group_2 _ Power_off _
    SECP_Restore RNP_Win_Restore"

#DrawPcx "boot.pcx"
DrawGIF "boot.gif"

# --
# --  edit the actions at the end of the file
# --
Set CurrentItem=1
LoadFont "menu.fnt"

:menu
Set exitmenu=""
DrawWindow $X $Y 200 180
DrawText ($X + 10)  ($Y+2) "Boot Selection" White
Set i=1
:loop
Set j=($i - 1)
DrawText ($X + 10)  (($Y+10) +($i*15)) "$MenuItem"{$j}/_= / White
Set i=($i+1)
if $i <= $MenuNum goto loop

:loop2
DrawBar ($X + 5) ( ($Y +10) + ($CurrentItem*15) ) 180 15 Yellow
Set j=($CurrentItem - 1)
DrawText ($X + 10) ( ($Y +10) + ($CurrentItem*15) ) "$MenuItem"{$j}/_= / Black
# hide cursor
TextAttr White White
GetKey Key
# Up key
if "$key" == " " if $CurrentItem > 1 Set CurrentItem = ($CurrentItem - 1)
# Down key
if "$key" == " " if $CurrentItem < $MenuNum Set CurrentItem = ($CurrentItem + 1)
# Enter
if "$key" == "\r" Set ExitMenu = "OK"

if "$ExitMenu" != "OK" goto menu

# --------------------
# now do what you have
# Edit below
# --------------------

```

```
51 if $CurrentItem == 1 goto rnplinux
52 if $CurrentItem == 2 goto rnpwinboot
53 if $CurrentItem == 3 goto secp1
54 if $CurrentItem == 4 goto secp2
55 if $CurrentItem == 5 goto dummy
56 if $CurrentItem == 6 goto poweroff
57 if $CurrentItem == 7 goto knoppix
58 if $CurrentItem == 8 goto secprestore
59 if $CurrentItem == 9 goto rnpwinrestore
60
61 :rnplinux
62 Set NAME="$BOOTP-Host-Name"
63 Set ADDR="$BOOTP-Your-IP"
64 Set TMP="$BOOTP-Your-IP"/.= /
65 Set PCNAME="$TMP"{3}
66 Set NFS="192.168.215.10"
67 Set GATE="192.168.215.10"
68 Set DNS="192.168.215.10"
69 Set MASK="255.255.255.240"
70
71 Set INIT="nfsroot=${NFS}:/export/rnp/root/ init=/rd/bin/init0"
72 Set IP="ip=${ADDR}:${NFS}:${GATE}:${MASK}:${NAME}::"
73 Set ARGS="$INIT $IP SERVER=$NFS nomodules"
74
75 linuxboot "rnplinux.krn" "$ARGS"
76 goto menu
77
78 :rnpwinboot
79 #boot cached... Windows
80 Set TMP="$BOOTP-Your-IP"/.= /
81 Set TEMP="$TMP"{3}
82 Set RNP="RNP"
83 Set PCNAME="$RNP$TEMP"
84 patch "autoexec.ref" "{:1}autoexec.bat"
85 patch "patch.ref" "{:1}temp/patch.reg"
86 hidebootprom
87 hdboot :1
88 goto menu
89
90
91 :secp1
92 #boot Linux ITSec1
93 DrawWindow 150 200 400 160 "Booting Linux.... Please wait."
94 TextAttr White LightGray
95 At 15,20 print "This may take a few seconds, be patient..."
96 # kernel 1 boot
97 copy "suse.krn" "{:2}itsec1/vmlinuz"
98 # crypto initrd
99 copy "cinitrd.gz" "{:2}cinitrd.gz"
100 # lilo files
101 copy "boot.b" "{:2}boot.b"
102 copy "chain.b" "{:2}chain.b"
103 copy "boot.msg" "{:2}boot.msg"
104 linuxboot "{:2}itsec1/vmlinuz" "root=/dev/ram0 init=/itsec1 rw" "{:2}cinitrd.gz"
105 goto menu
106
107 :secp2
108 #boot Linux ITSec2
109 DrawWindow 150 200 400 160 "Booting Linux.... Please wait."
110 TextAttr White LightGray
111 At 15,20 print "This may take a few seconds, be patient..."
112 # kernel 2 boot
113 copy "suse.krn" "{:2}itsec2/vmlinuz"
114 # crypto initrd
115 copy "cinitrd.gz" "{:2}cinitrd.gz"
116 # lilo files
117 copy "boot.b" "{:2}boot.b"
118 copy "chain.b" "{:2}chain.b"
119 copy "boot.msg" "{:2}boot.msg"
120 linuxboot "{:2}itsec2/vmlinuz" "root=/dev/ram0 init=/itsec2 rw" "{:2}cinitrd.gz"
121 goto menu
122
123 :dummy
124 # " "
125 goto menu
```

```
126
127 :poweroff
128 Poweroff
129 goto menu
130
131
132 :knoppix
133 #Booten von CD
134 Set CacheNever="ON"
135 LoadRamDisk "boot-de.ima"
136 HideBootProm
137 FloppyBoot
138
139 :secprestore
140 #Boot Restoresystem
141 linuxboot "install2418.krn" "root=/dev/nfs ro nfsroot=192.168.100.100:/backup/restore
      ip=dhcp"
142 goto menu
143
144 :rnpwinrestore
145 #boot Windows
146 #Set CacheAlways="on"
147 #Set CacheNever="off"
148 Set TMP="$BOOTP-Your-IP"/.= /
149 Set TEMP="$TMP"{3}
150 Set RNP="RNP"
151 Set PCNAME="$RNP$TEMP"
152 fullunzip "win98.imz" 1
153 patch "autoexec.ref" "{:1}autoexec.bat"
154 patch "patch.ref" "{:1}temp/patch.reg"
155 hidebootprom
156 hdboot :1
157 goto menu
```

Listing B.2: Backup/Restoresytem Serverseitige Konfigurationsdatei

```
1  #!/usr/bin/perl -w
2  ##########
3  # Config #
4  ##########
5
6
7  $num_slots = 5;
8  $prefix = "/backup";
9  $secpserver = "192.168.216.254";
10 $secpserverrest = "192.168.100.100";
11
12 $u_action = "";
13 $u_pc = "";
14 $u_slot = "";
15 $u_group = "";
16 $u_ip = "";
17 $timeserver = "$secpserverrest";
18 $privkey = "$prefix/keys/restore_priv/id_dsa";
19 $privkeyuser = "$prefix/keys/restore_priv/id_dsa.user";
20 $pubkeydir = "$prefix/keys/restore_pub";
21 $pooldir = "$prefix/pool"; # server
22 $tmpdir = "$prefix/tmp"; # server
23 $customfiles = "$prefix/myfiles"; # server
24 $sshpath = "/usr/bin/ssh -l root -i $privkey"; # server
25 $u_sshpath = "/usr/bin/ssh -l root -i $privkeyuser"; # server
26 $scppath = "/usr/bin/scp -i $privkey"; # server
27 $greppath = "/usr/bin/grep"; # server
28 $gzippath = "/usr/bin/gzip"; # server
29 $catpath = "/bin/cat"; # server
30 $arppath = "/sbin/arp"; # server
31 $expectpath = "/usr/bin/expect"; # server
32 $sfdiskpath = "/usr/sbin/sfdisk"; # client
33 $genkey = "head -c 45 /dev/random | uuencode -m - | head -2 | tail -1"; #server
34 $gpgpath = "/usr/bin/gpg"; # server
35 $ntpdatepath = "/usr/sbin/ntpdate"; # client
36 $hwclockpath = "/sbin/hwclock"; # client
37
38
39 $mkswap = "/sbin/mkswap"; # client
```

```
$mkdos = "/sbin/mkdosfs"; # client
$mkext2 = "/sbin/mke2fs"; # client
$noformat = "/bin/false"; # client
$instmp = "/install";
$scriptdir = "./scripts";
$instmptmp = "/install2";
$myhd = "/dev/hda";
$tarpath = "/bin/tar"; # client
$dumppath = "/sbin/dump.static"; # client
$restorepath = "/sbin/restore.static"; # client
$mountpath = "/sbin/mount"; # client
$unmountpath = "/sbin/umount"; # client
$rmpath = "/bin/rm"; # client
$mkdirpath = "/bin/mkdir"; # client
$cppath = "/bin/cp"; # client
$losetuppath = "/sbin/losetup"; # client
$defaultfstab = "/dev/cdrom      /media/cdrom    auto    ro,noauto,user,exec 0 0\
    ndevpts  /dev/pts        devpts  defaults 0 0\n/dev/fd0        /media/floppy
      auto    noauto,user,sync 0 0\nproc    /proc   proc    defaults 0 0\nusbdevfs
           /proc/bus/usb   usbdevfs        noauto 0 0\n";


# partitions: size, type, bootable, mkfs options(journaling, ...), defaultimage,
    postinstall script
$partitions{"1"} = [1500, "b",  "*",""];
$partitions{"2"} = [200,  "6",  "", ""]; # bpbatch hat mit ext2 probleme :/
$partitions{"3"} = [18000,"5",  "", ""];
$partitions{"4"} = [0, 0, "",   "", ""];
$partitions{"5"} = [512,  "82", "", ""];
$partitions{"6"} = [8000,"83", "", "-j"];
$partitions{"7"} = [512,  "82", "", ""];
$partitions{"8"} = [8000,"83", "", "-j"];

$mkfs{"6"} = "$mkdos";
$mkfs{"83"} = "$mkext2";
$mkfs{"82"} = "$mkswap";

# partitions which are important for the specific groups
$groups{"itsec1"} = ["2", "3", "5", "6"];
$groups{"itsec2"} = ["2", "3", "7", "8"];

$user_pw = '1234';
$master_pw = '7001337';



# root partition, swap partition, defaultimage
$install{"itsec1"} = ["6", "5", "itsec1_default.gz"];
$install{"itsec2"} = ["8", "7", "itsec2_default.gz"];

$macs{"00:04:76:DB:FA:40"} = "pcsec1"; #
$macs{"00:04:76:DA:C1:FD"} = "pcsec2";
$macs{"00:04:76:DB:FA:28"} = "pcsec3"; #
$macs{"00:04:76:DA:C1:11"} = "pcsec4"; #
$macs{"00:04:75:BF:F7:E5"} = "pcsec5"; #
$macs{"00:04:76:DB:FA:6A"} = "pcsec6"; #
$macs{"00:04:76:DA:C1:03"} = "pcsec7"; #
$macs{"00:04:76:DA:C2:3F"} = "pcsec8"; #
$macs{"00:04:76:DB:FA:37"} = "pcsec9"; #
$macs{"00:04:76:DB:FA:32"} = "pcsec10"; #

#  gruppen + passwörter
$secpauth{"pcsec1-itsec1"} = "tasdfFest";
$secpauth{"pcsec1-itsec2"} = "mi5hFch";
$secpauth{"pcsec2-itsec1"} = "tesgDFst";
$secpauth{"pcsec2-itsec2"} = "m2iDch";
$secpauth{"pcsec3-itsec1"} = "tegnGst";
$secpauth{"pcsec3-itsec2"} = "mbicxGh";
$secpauth{"pcsec4-itsec1"} = "teDasdfst";
$secpauth{"pcsec4-itsec2"} = "miVaGch";
$secpauth{"pcsec5-itsec1"} = "teDgysFFt";
$secpauth{"pcsec5-itsec2"} = "mias453dfch";
$secpauth{"pcsec6-itsec1"} = "te42asdfast";
$secpauth{"pcsec6-itsec2"} = "fggmfsaich";
$secpauth{"pcsec7-itsec1"} = "tasdfeasdyyt";
```

```
$secpauth{"pcsec7-itsec2"} = "fm334ch";
$secpauth{"pcsec8-itsec1"} = "sadt23151st";
$secpauth{"pcsec8-itsec2"} = "fdgh343ch";
$secpauth{"pcsec9-itsec1"} = "ht4527nbt";
$secpauth{"pcsec9-itsec2"} = "mhsa3dfch";
$secpauth{"pcsec10-itsec1"} = "easdagsdft";
$secpauth{"pcsec10-itsec2"} = "m435asdfch";
```

Listing B.3: ssh config auf Serverseite

```
# This is the ssh client configuration file.  See
# ssh_config(5) for more information.  This file provides defaults for
# the user, and the values can be changed on the command line.

Host *

CheckHostIP no
StrictHostKeyChecking no
```

Listing B.4: Start Script für Notfallrestore

```
#!/bin/sh
ssh 192.168.100.100 -l secpback
```

Listing B.5: DHCP Konfigurationsdatei

```
ddns-update-style none; ddns-updates off;

default-lease-time              600;
max-lease-time                  7200;
server-identifier 192.168.100.100;

option routers 192.168.216.194;
option domain-name "secp.nm.informatik.uni-muenchen.de";
option domain-name-servers 192.168.216.254;
option space bpbatch;
option bpbatch.script code 135 = text;
option bpbatch.vendorclass code 60 = text;
option bpbatch.vendoroptions code 43 = string;

authoritative;

subnet 192.168.100.0 netmask 255.255.255.0 {
use-host-decl-names on;

  filename "bpbatch";
  option broadcast-address 192.168.100.0;
  option bpbatch.vendorclass = "PXEClient";
  option bpbatch.vendoroptions = 01:04:00:00:00:00;
  range 192.168.100.11 192.168.100.21;

# Router Maschinen werden über knoppix cd gebootet
# Konfiguration über dhcp
# Konfiguration der Routen über ein Skript
# config_<IP_Addresse> im Directory
# /tftpboot/remote_config

host bootsec1 {
      hardware                ethernet 00:04:76:DB:FA:40;
      fixed-address           192.168.100.31;
 }

host bootsec3 {
      hardware                ethernet 00:04:76:DB:FA:28;
      fixed-address           192.168.100.33;
 }


 host bootsec5 {
      hardware                ethernet 00:04:75:BF:F7:E5;
      fixed-address           192.168.100.35;
 }

```

```
48
49   host bootsec7 {
50       hardware                ethernet 00:04:76:DA:C1:03;
51       fixed-address           192.168.100.37;
52   }
53
54   host bootsec9 {
55       hardware                ethernet 00:04:76:DB:FA:37;
56       fixed-address           192.168.100.39;
57   }
58
59
60 }
```

Literaturverzeichnis

[Crba02] BARRATT, CRAIG: *Backup PC Introduction*, 2 Aug 2002, http://belnet.dl.sourceforge.net/sourceforge/backuppc .

[Pre99] PRESTON, W. CURTIS: *Unix Backup and Recovery*. O'Reilly and Associattes, November '99. 730 p.

[Siwa03] KERN SIBBALD, JOHN WALKER: *The Bacula Storage Management System Version 1.30*, 23 February 2003, http://www.bacula.org/html-manual/index.html .

[Stcl00] MARC VUILLEUMIER STÜCKELBERG, DAVID CLERC: *Linux Remote-Boot mini-HOWTO*, February 2000, http://cui.unige.ch/info/pc/remote-boot/howto.html . 40 p.