INSTITUT FÜR INFORMATIK

DER LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Erstellung einer Knoppix Boot CD und Upgrade der Infrastruktur für das Praktikum IT-Sicherheit

Manuel Tundo

Aufgabensteller: Prof. Dr. H. G. Hegering

Betreuer: Dr. Helmut Reiser

Abstract. Ziel dieses Forgeschrittenenpraktikums ist es, eine Knoppix-Live-CD zu erstellen, durch die fehlende Teilnehmer des IT-Sicherheit-Praktikums ersetzt werden können. Die CD konfiguriert alle benötigten Dienste selbständig. Ausserdem soll die Umgebung des Praktikums IT-Sicherheit von Suse 8.0 auf Suse 9.1 aktualisiert werden. Das bestehende Backup-/Restore-System muss dabei beibehalten werden.

Inhaltsverzeichnis

1 Einleitung

Im Verlauf dieses Fortgeschrittenenpraktikums wurden im ersten Schritt die Rechner des IT-Sicherheit-Praktikums [1] von SuSE 8.0 auf SuSE 9.1 [2] aktualisiert. Im zweiten Schritt wurde die bekannte Knoppix-LiveCD [3] remastert und an die Bedürfnisse des IT-Praktikums angepasst.

1.1 Motivation

Im IT-Sicherheitspraktikum, das jedes Semester an der Lehr- und Forschungseinheit Prof. Hegering angeboten wird, arbeiten meist 2 Gruppen im Team zusammen. Jede Gruppe, bestehend aus 2 Personen, hat ihren eigenen Rechner, an dem sie die Versuche bearbeiten muss. Das Praktikum nutzt je nach Versuchstag entweder eine Hub- oder eine Sterntopologie. Letztere wird in den meisten Versuchen verwendet. Dabei dient der Rechner eines Teams als Router und ist somit Vorraussetzung für das erfolgreiche Arbeiten der Gruppe, die am Rechner hinter dem Router sitzt. Zur Veranschaulichung soll Abb. 1 dienen:

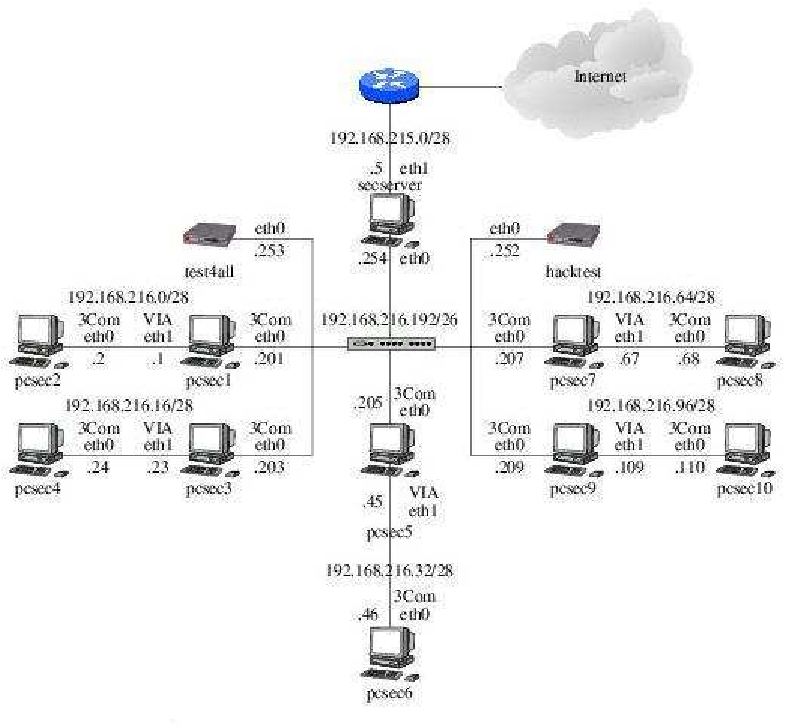

Abb. 1 Sterntopologie

Da also immer 2 Gruppen (teilweise sehr eng) zusammenarbeiten müssen, ist es umso verhängnisvoller, wenn die Gruppe, die am Routerrechner arbeitet, fehlt: Die anwesende Gruppe kann Teile oder sogar den gesamten Versuch nicht durchführen. Beispielsweise ist dies beim VPN-Versuch der Fall, in dem eine verschlüsselte Verbindung zwischen den beiden Teamrechnern aufgebaut werden muss. Ebenso beim DNS-Versuch, bei dem der Routerrechner als DNS-Forwarder für den Partnerrechner dient. Um dieses Problem zu beheben, wurde im Rahmen dieses Fortgeschrittenenpraktikums die KNOPPIX-LiveCD remastert mit dem Ziel, fehlende Praktikumsgruppen zu ersetzen und den anwesenden Gruppen einen reibungsfreien Arbeitsablauf zu garantieren. KNOPPIX ist eine Debian-basierende GNU/Linux-Distribution, die ohne Installation auskommt und direkt von CD gebootet werden kann.

Zusätzlich sollten alle Rechner des Praktikums auf die zum Zeitpunkt der Forgeschrittenenpraktikums aktuelle Version SuSE 9.1 aktualisiert werden.

1.2 Aufgabenstellung

Dieses Fortgeschrittenenpraktikum ist in zwei Aufgaben unterteilt:

- Update der Rechner im IT-Sicherheitspraktikum auf SuSE 9.1
- Remastern einer KNOPPIX-LiveCD [4]

Das Update wird zuerst in einer kleinen Testumgebung ausgeführt, bevor alle Rechner incl. dem Hauptserver (Secserver, der zentrale Rechner, von dem alle Clients booten und auf dem die VLAN-Konfiguration enthalten ist) aktualisiert werden. Verläuft alles in der Testumgebung problemlos, dann wird das gesamte Praktikum aktualisiert.

Das Remastern der Knoppix-Distribution nimmt den anderen Teil des Fortgeschrittenenpraktikums in Anspruch. Es soll eine auf Knoppix basierende LiveCD entwickelt werden, die es ermöglicht von ihr zu booten, während dessen zu erfragen, in welchem Rechner sie gebootet wird und welche Dienste benötigt werden. Anschliessend werden die benötigten Konfigurationen und Einstellungen dynamisch vorgenommen. Die CD soll somit ein Ersatz für die am Routerrechner arbeitende Gruppe darstellen, so dass die Partnergruppe trotz dem Fehlen dieser Gruppe alle Versuche erfolgreich durchführen kann.

1.3 Inhaltsüberblick

Der erste Teil ist dem Rechnerupgrade gewidmet. Es wird beschrieben, wie die Testumgebung aufgebaut ist, wie die nötigen Images erstellt werden und wie das Boot-/Restorekonzept aussieht.

Im zweiten Teil wird Knoppix mit seinem (Boot)Konzept vorgestellt. Anhand der Praktikums-Anforderungen werden verschiedene Realisierungsmöglichkeiten entworfen, diskutiert und realisiert. Abschließend gibt es eine kleine “Bedienungsanleitung” zum Arbeiten mit der LiveCD und eine Beschreibung der vorgenommenen Änderungen.

2 Upgrade der Praktikumsrechner

2.1 Konzeption

Bevor alle Rechner auf SuSE 9.1 aktualisiert werden, wird das Szenario zuerst in einer aus 4 Rechnern bestehenden Testumgebung durchgeführt. Sollte es später keine Probleme geben, so wir das gesamte Praktikum aktualisiert.

Einer der 4 Rechner dient als Server, auf dem später das Image einer SuSE 9.1-Installation liegt. Die anderen Rechner sind die Clients, die das Image vom Server bekommen und es installieren. Die genaue Funktionsweise wird in Abschnitt 2.2 erläutert. Eigentlich war geplant, dass der als Server dienende Rechner über die Funktion *SuSE Upgrade* im YAST-Menü aktualisiert wird. Jedoch funktionierte das Update auf diese Weise nicht, weshalb das System komplett neu aufgesetzt werden musste. Nachdem der als Server dienende Rechner aktualisiert worden ist, muss auf ihm ein DHCP-Server aufgesetzt werden. Anschließend wird der Bootloader *bpbatch*, der das Booten über das Netzwerk ermöglicht, und ein TFTP-Server installiert. Ebenfalls muss ein NFS-Server eingerichtet werden, damit die Clients das Restore-System über das Netz mounten können.

Die Clients werden neu augesetzt, indem man im Bootmenü den Punkt "IT-Sicherheit-Restore" ausführt. Dadurch wird das auf dem NFS-Server befindliche Restoresystem gebootet.

2.2 Durchführung

Der Rechner pcsec6 wird zum Server umfunktioniert. Dazu muss zuerst SuSE9.1 installiert werden. Leider funktioniert die von SuSE bereitgestellte Funktion *SuSE Update* nicht: Es wird als Quellmedium die CD angegeben, doch leider wird die CD nicht erkannt. Es wird *"Unable to mount filesystem"* gemeldet. Auch das Kopieren der CDs auf Festplatte half ebenso wenig wie ein mounten über NFS. Deshalb musste die Installation manuell über eine *Neuinstallation* gemacht werden. Dazu wird eine bestehende Partition verwendet.

Auf dem neu aufgesetzten System wird ein DHCP-Server eingerichtet. Er dient dazu, den Clients zuerst eine IP, und dann den Bootloader *Bpbatch* zur Verfügung zu stellen. Der Bootloader wird über das TFTP-Protokoll, ein minimales FTP, zum Client übertragen. Dazu muss das Verzeichnis */tftpboot* existieren. Darin liegen der Bootloader mit seiner Konfigurationsdatei *bpbatch.bpb*, der zu bootende Kernel und die *Initial Ramdisk*. Die *initrd* ein kleines filebasiertes Loopback-Filesystem, das in eine Ramdisk geladen wird. Dies geschieht unmittelbar nach Laden des Kernels, aber noch vor dem Starten des eigentlichen Betriebssystems.

Bpbatch wird auf dem Client ausgeführt. Ein Restore des Clients, und damit ein Update des Betriebssystem, erfolgt über den Punkt *"ITSEC Restore"* im Bootmenü. Dabei wird vom Bootloader ein Kernel, der auf dem Server bereitsteht, auf den Client kopiert und geladen. Als Root-Filesystem dient das über NFS exportierte Verzeichnis */backup/restore*, in dem ein entpacktes tar-Image des laufenden Systems liegt. Das Image wird mit

```
tar -czf /backup/restore/backup.tgz --exclude /tmp --exclude /home
--exclude /proc --exclude /tftpboot --exclude /backup --exclude
/usr/src /
```

erstellt. Es muss entpackt und angepasst werden. Z.B. müssen alle Client-spezifischen Konfigurationsdateien in *etc* angepasst werden. Dazu gehört der Hostname, die Netzkonfiguration (Namensauflösung, Routen, IP-Adressen) und das Entfernen überflüssiger Dienste aus den Runlevel-Verzeichnissen *etc/rc*.d.* Das System soll nur minimal starten, also sind Dienste wie z.B. Squid und DNS aus den einzelnen Runlevels zu entfernen. Als Defaultrunlevel wird in der */etc/inittab* Runlevel 3 eingestellt. Ausserdem müssen die Verzeichnisse */proc*, */install* und */install2* angelegt werden, da diese später vom Systerm und den Restore-Skripten benötigt werden.

Die IP des Testservers wird auf die des Secservers gesetzt, am Secserver selbst wird das Netzwerk über */etc/init.d/network stop* angehalten, um IP-Konflikte zu vermeiden. Damit wird gewährleistet, dass die am Testserver eingerichtete Konfiguration auch auf dem Secserver funktioniert.

Der Kernel und die Initial Ramdisk müssen in das */tftpboot*-Verzeichnis kopiert werden. In der Konfigurationsdatei *bpbatch.bpb* muss der Dateiname des Kernels angepasst werden. Während des Fortgeschrittenenpraktikums kam es zu dem Problem, dass der Bootvorgang auf den Clients mit folgender Fehlermeldung abgebrochen hat:

Uncompressing Linux...

Invalid Compressed Format (err=1)

Es stellte sich heraus, dass der Netzbootloader ausschliesslich einen Kernel laden kann, der kleiner als 960kB ist, ansonsten bricht der Bootvorgang wie beschrieben ab. Da der neue SuSE9.1-Kernel ca.1,3MB gross ist, musste er neu kompiliert werden. Nur die wichtigsten Treiber, wie die für das Filesystem, Netzwerk und Crypto-Optionen werden fest in den Kernel einkompiliert, alles andere wird als Modul ausgelagert, um die Grösse von 960kB nicht zu überschreiten.

Zum erfolgreichen Booten der Clients nach dem Restore, muss die Initial Ramdisk angepasst werden. Dazu wird die Datei *cinitrd.gz* entpackt und daraufhin *loop*-gemountet:

```
#gunzip cinitrd.gz
#mount cinitrd /mnt/initrd -o loop
```

Die Module im Verzeichnis */mnt/floppy/lib/modules* müssen aktualisiert werden. Dazu ersetzt man die Moduldateien durch die aktuellen Versionen aus dem */lib/modules*-Verzeichnis. Nach den Änderungen wird die Initial Ramdisk ausgehängt und wieder gepackt.

```
#umount /mnt/initrd
#gzip cinitrd
```

Das Backup-/Restore-Skript, das bereits in einem anderen Fortgeschrittenenpraktikum entwickelt wurde, benötigt die User *secpback* und *secpuser* auf dem Server. Also müssen sie angelegt und angepasst werden: Statt wie im Normalfall eine Shell, wird nach dem Login eines der Benutzer das Backup-/Restore-Skript ausgeführt. Diese Skripte wurden im Rahmen des Fortgeschrittenenpraktikums *"Konzeption und Realisierung einer sicheren Multiboot- und Backup-Infrastruktur"* [9] entwickelt. Damit sie auch sofort nach dem Login abgearbeitet werden, muss */etc/passwd* angepasst werden:

```
secpback:x:501:100:secpbackup:/backup:/backup/master.pl
secpuser:x:502:101:userbackup:/backup:/backup/user.pl
```

Ebenfalls muss im Restore-System der User *install* existieren und angelegt werden. Zum Abschliessen der Serverkonfiguration wird in */backup* die von den Skripten benötigte Verzeichnisstruktur angelegt. Dazu gehören u.a. die Verzeichnisse *myfiles* und *pool*, die von den Backup-Skripten des Multiboot-Fopras verwendet werden. Am besten werden diese Dateien und Verzeichnisse vom Secserver kopiert.

Nachdem ein Client das Restore-System gebootet hat, wird er zum Login aufgefordert. Als User *install* loggt man sich ein, was die Ausführung eines kleinen Shell-Skripts bewirkt, das den User *secpback* über SSH am Server anmeldet. Nach dem Login von *secpback* wird das Skript *master.pl* im */backup*-Verzeichnis interaktiv abgearbeitet. Um während der Backup-/Restore-Prozedur nicht ständig das root-Passwort eingeben zu müssen, ergänzt man die Datei *authorized-keys* im *.ssh*-Verzeichnis des Users *root* im Restore-System um den Public-Key des Users *root* vom Secserver.

Wählt man den Restore des Defaultimages, so werden auf dem Client die Partitionen mit einem Cryptofilesystem angelegt und das Defaultimage des SuSE9.1-Systems installiert. Das Restore-System nutzt eine Kombination aus *Dump* und *Restore*. Dazu wird mittels

```
#dump -0anf /backup/myfiles/itsec1_default.dump -e 17386,21547,165 /
```

ein Dump erzeugt und anschliessend mit

```
#gzip /backup/myfiles/itsec1_default.dump > /backup/myfiles/itsec1_default.gz
```

komprimiert. Die Optionen des Dump-Aufrufs sind im Folgenden erklärt:

- ***0***: Dump-Level. Level 0 entspricht einem vollständigem Backup.
- ***a***: auto-size, schreibt alles in den Dump bis ein end-of-media gemeldet wird
- ***n***: erforderliche Userinteraktionen erfolgen automatisch
- ***f datei***: schreibe den Dump in *datei*

- ***e inodes:*** der Dump beinhaltet nicht die als Parameter angegebenen Inodes/Verzeichnisknoten. Die zu einem Verzeichnis gehörende Inode-Nummer lässt sich mit einem Aufruf von *stat verzeichnisname* herausfinden. Im Beispiel handelt es sich um die Verzeichnisse */proc, /backup* und */dev.*

Dieser Dump wird beim Restore entpackt und auf dem Client installiert.

2.3 Probleme

Bei der Durchführung traten Probleme mit der Verschlüsselung der Partitionen auf. Das Backup-Restore-System partitioniert selbstständig die Festplatte. Nachdem die Partitionen mit AES verschüsselt werden, wird der zuvor erstellte Dump eingespielt. Schliesslich werden alle Filesysteme ausgehängt.

Beim nächsten Boot sollten die Clients in der Lage sein, die verschlüsselten Partitionen mit Hilfe eines Passwortes enschlüsseln können. Das war jedoch nicht der Fall. Hier der genaue Ablauf der Verschlüsselung:

```
losetup -e AES128 /dev/loop0 /dev/hdxx
mkfs -t ext2 /dev/loop0
# Image einspielen
losetup -d /dev/loop0
```

Beim Versuch die verschlüsselte Partition nach dem Aushängen wieder zu mounten, schlug leider ohne Fehlermeldungen fehl. Wie sich herausstellte, ist dies ein SuSE-spezifisches Problem, da SuSE eine eigene Verschlüsselung mitliefert, die fest in den Kernel integriert ist. Als Abhilfe wurde der Kernel neu kompiliert und darauf geachtet, dass unter "Blockdevices" die Option »Loopback device support« auf m wie Modul gesetzt ist. Ausserdem wurden *losetup* und *mount* gepatcht, da diese Tools in der älteren Version fehlerhaft sind. Trotz dieser Änderungen und Bugfixes konnte das Problem nicht behoben werden. Auch durch Posten in verschiedene Mailinglisten konnte keine Lösung gefunden werden. Im Übrigen besteht dieses Problem bei SuSE9.0 nicht.

Damit ist die Migration von SuSE8.0 auf SuSE9.1 leider nicht erfolgreich. Alle vorgenommenen Änderungen an der Praktikumsumgebung wurden rückgängig gemacht. Somit wurde die alte SuSE8.0-Umgebung beibehalten.

3 Remastering KNOPPIX

3.1 Was ist KNOPPIX?

Ein Auszug von *www.knopper.net/knoppix*:

> *KNOPPIX ist eine komplett von CD lauffähige Zusammenstellung von GNU/Linux-Software mit automatischer Hardwareerkennung und Unterstützung für viele Grafikkarten, Soundkarten, SCSI- und USB-Geräte und sonstige Peripherie. KNOPPIX kann als Linux-Demo, Schulungs-CD, Rescue-System oder als Plattform für kommerzielle Software-Produktdemos angepasst und eingesetzt werden. Es ist keinerlei Installation auf Festplatte notwendig. Auf der CD können durch transparente Dekompression bis zu 2 Gigabyte an lauffähiger Software installiert sein.*

Also eine GNU/Linux-Distribution, die bei Bedarf ins CDROM gelegt wird und ohne Installation sofort nutzbar ist.

3.2 Ziele und Anforderungen

Ziel dieses Teils des Fortgeschrittenenpraktikums ist es, dass ein Fehlen einer Gruppe während dem IT-Praktikum keine Auswirkungen auf die Partnergruppe hat.

In den meisten Versuchen müssen die Gruppen zweier Rechner zusammenarbeiten, deshalb ist es unter Umständen für ein Team fatal, wenn die Partnergruppe nicht anwesend ist. Die Knoppix-LiveCD soll speziell an das Praktikum angepasst werden, so dass sie die Funktionen der am Router-Rechner arbeitenden Gruppe vollständig übernimmt. Zu diesen Funktionen des Routers gehören:

- Routing
- Firewall
- Network Address Translation (NAT)
- DNS-Forwarder
- SSH-/Telnet-Server
- Mailserver
- VPN (IPSec)

Es soll von der Live-CD gebootet werden und dabei macht der User Eingaben, die es dem System ermöglichen, sich selbstständig zu konfigurieren.

Es müssen zwei unterschiedliche Konfigurationen berücksichtigt werden: die Hub- und die Sterntopologie [6].

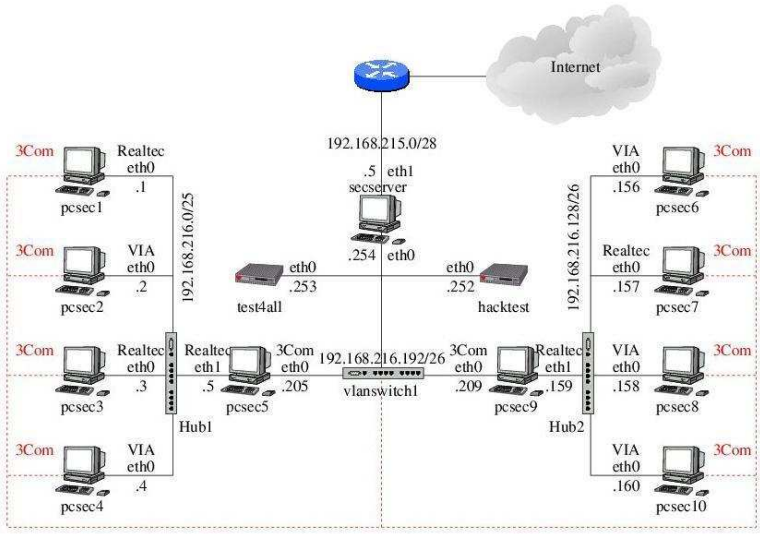

Abb. 2 Hubtopologie

Man muss zusätzlich unterscheiden, in welchem Rechner die CD gebootet wird, damit eine einwandfreie Eingliederung des Rechners in das Praktikum möglich ist (vgl. Abb. 2). Die Basis ist also eine funktionierende Netzwerkkonfiguration (IP-Adressen/Subnetzmasken, Routing). Der User soll auswählen können, welche Dienste benötigt werden und welche nicht. Die gewählten Dienste sollen dynamisch konfiguriert werden, so dass der Gruppe hinter dem Router ein reibungsfreies Arbeiten ermöglicht wird.

3.3 Lösungsmöglichkeiten

Es gibt verschiedene Möglichkeiten, mit denen die Ziele und Anforderungen erfüllt werden können. Im Folgenden werden zwei beschrieben:

3.3.1 Cheatcodes

Cheatcodes nennt man Bootparameter, die den Standardbootoptionen mitgegeben werden können. So z.B. *nodhcp*, der die automatische Konfiguration der Netzwerkinterfaces über das DHCP-Protokoll verhindert.

Es wäre also möglich, dass man für jeden Dienst einen Bootparameter erstellt, der angegeben werden muss, falls der Dienst gestartet werden soll. Wird das System beispielsweise mit den Bootparametern *"SQUID=YES FIREWALL=NO"* gestartet, steht dies auch in der */proc/cmdline*. Die aktuelle IP vom Rechner muss ebenfalls als Bootparameter übergeben werden. Man muss also mit einem entsprechenden Skript prüfen, ob der Squid-Proxy gestartet werden soll oder nicht. Bsp:

```
if [ $(grep "SQUID=YES" /proc/cmdline) -eq 1 ];then /etc/init.d/squid start
```

Diese Methode verlangt vom Anwender, dass er sich mit den möglichen Bootparametern auseinander setzen muss, was ihn Zeit kostet und ihm unter Umständen Probleme bereitet. Aber dafür ist sie recht flexibel in ihrer Handhabung. Zur Realisierung müssen diese Cheatcodes dem System bekannt gemacht werden, was durch Einträge in die */etc/init.d/knoppix-autoconfig* vorgenommen wird.

3.3.2 Erweitertes Bootmenü

Eine der vorigen Methode sehr ähnliche Lösung besteht darin, das vorhandene Bootmenü zu erweitern. Es muss für jeden Versuchstag/Versuchsaufgabe und für jeden Routerrechner ein Eintrag vorhanden sein. Das ist auch gleichzeitig der Hauptnachteil. Vorteil ist zwar, dass sich der Anwender nicht mit den Cheatcodes auseinandersetzen muss, aber der Nachteil wiegt sicherlich schwerer: Das Bootmenü wird extrem unübersichtlich. Deshalb kommt diese Lösungsmöglichkeit nicht in Frage.

3.3.3 knoppix.sh

KNOPPIX führt gegen Ende des Bootvorgangs ein Skript names *knoppix.sh* im Verzeichnis */etc/init.d* aus. Durch dieses Skript lassen sich wiederum andere (eigene) Skripte anstossen. Dies führt zu einer weiter Lösungsmöglichkeit.

Man ändert die Standardbootoptionen so ab, dass keine DHCP-Abfrage vorgenommen wird. Dann wird ein Skript durchlaufen, das den Anwender nach der IP und Subnetzmaske des Rechners fragt, in dem sich die CD befindet. Anschließend hat er die Möglichkeit aus einem Menü zu wählen, welche Dienste er gestartet haben will. Das Ergebnis dieser Abfragen ist die Konfiguration der Netzwerkinterfaces, des Routings, der gewählten Dienstkonfigurationsdateien und es werden die benötigten Services gestartet.

Vorteil dieser Methode ist, dass der Anwender eine sehr einfache und schnelle Möglichkeit hat, das System so zu starten, wie er es benötigt. Zudem ist es angenehm übersichtlich im Gegensatz zu einem überfüllten Bootmenü wie in Abschnitt 3.3.2 erläutert. Deswegen wird diese Lösungmöglichkeit im Fortgeschrittenenpraktikum realisiert.

3.4 Der Bootvorgang

Knoppix ist eine bootbare CDROM nach dem El-Torito Standard [7]. Als bootloader wird standardmäßig *Syslinux* verwendet. Die Aufgabe des Bootloaders ist es, den Kernel und die Initial Ramdisk zu laden. *Syslinux* benötigt jedoch Bootimages von genau 960KB, 1.44MB oder 2.88MB Grösse. Die meisten BIOS-Versionen behandeln nur 1.44MB-Images korrekt. Für die modifizierte Knoppix-CD reichen diese 1.44MB leider nicht aus, weshalb auf den Bootloader *Isolinux* gewechselt wird, bei dem die Images keine vorgegebene Grösse haben müssen. Die zugehörige Konfigurationsdatei heisst *isolinux.cfg.*

Die Initial Ramdisk erkennt durch einen Testmount das Bootmedium und lädt anschliessend das *cloop*-Modul, mit dessen Hilfe das *cloop*-Dateisystem, ein komprimiertes Loop-Dateisystem, gemountet wird. Das System wird nun durch ein Shellscript namens *knoppix-autoconfig* initialisiert:

- Einrichten einer zusätzlichen Ramdisk für /home und /var
- Hardware-Erkennung
- Starten der Dienste
- Automatische X-Serverkonfiguration
- einbinden von zusätzlicher (benutzerdefinierten) Konfiguration durch das Shellscript *knoppix.sh.*

Während dem Bootvorgang soll der User gefragt werden, welche IP-Adresse und Subnetzmaske der Rechner hat, in dem sich die CD befindet. Danach wird er nach den Diensten gefragt, die gestartet werden sollen. Realisiert wird dies wie folgt:
Der standardmäßig gestartete DHCP-Client wird mit der Bootoption *nodhcp* ausgeschaltet. Dies erfolgt durch Anpassung der *isolinux.cfg.*

Knoppix ruft gegen Ende des Bootvorgangs das Script *knoppix.sh* auf. Dieses wird um den Aufruf von eigenen Skripten ergänzt, die den User die benötigten Eingaben machen lassen. Mittels dem Tool *dialog,* das eine rudimentäre graphische Oberfläche auf der Konsole darstellen kann, werden die Usereingaben erfragt und in Variablen gespeichert, die wiederum in Textdateien abgelegt werden.

Nachdem die Benutzerinteraktion abgeschlossen ist, werden durch das soeben durchlaufene Skript andere, durch die Usereingaben abhängige, Skripten ausgeführt. Zuerst wird das gesamte Netzwerk inkl. Routing aktiviert. Den Netzwerkinterfaces werden also IP-Adresse und Subnetzmaske zugeordnet. Ausserdem wird das Routing für den Rechner eingerichtet.

Für jeden zur Auswahl stehenden Dienst gibt es ein passendes Skript, das diesen Service bei Bedarf startet. Soll beispielsweise die Firewall gestartet werden, so wird ein Skript angestoßen, das die aktuellen IP-Adressen und Subnetzmasken ausliest, dadurch die IP und Subnetzmaske des Partnerrechners bestimmen kann und letztendlich die Firewallkonfiguration passend zur aktuellen Konfiguration durchführt.

3.5 Remaster-Anleitung

3.5.1 Zutaten

Fogende *"Zutaten"* werden idealerweise benötigt, um Knoppix zu remastern:

- Eine Knoppix-CD.

- 1GB Arbeitsspeicher; falls man weniger hat, muss eine SWAP-Partition angeleget werden.
- ein Rechner, auf dem ein installiertes Linux läuft und 2 Partitionen:
  Entwicklungspartition, auf der die Knoppix-CD entwickelt wird, mit mindestens 4GB freien Platz (bei weniger als 1GB RAM werden 5GB benötigt)
  Install-Partition ist eine leere 750MB-Partition.
- Das Kommandozeilentool *create_ compressed_ fs*, das von der Knoppix-CD kopiert werden kann.
- Eine *kick-list* [8]. In dieser Liste stehen die Namen der Pakete, die ohne Verletzen von Abhängigkeiten entfernt werden können.

Für das Praktikum wurde die Knoppix-c't-Edition verwendet, da sie sowohl den 2.4er-Kernel als auch den neuen Kernel 2.6.1 verwenden kann. Da im IT-Sicherheitspraktikum ein VPN mit *ipsec* eingerichtet werden muss, wird ein Kernel benötigt, der *FreeS/WAN* beinhaltet. Dazu muss ein Kernel mit einer Version kleiner als 2.6 gepatched werden, im Kernel ab Version 2.6 ist FreeS/WAN bereits integriert. Die fertige remasterte CD nutzt Kernel 2.6.1.

3.5.2 Emulieren der CD auf Festplatte

Schritt 1: Anlegen einer Partition

Legen Sie eine mindestens 750MB grosse Partition an. Alle folgenden Schritte werden auf der Install-Partition ausgeführt (hda1)

Schritt 2: Verzeichnisstruktur anlegen:

1. Mounten Sie die neue Partition und legen Sie ein Verzeichnis *harddrive_ boot* an.

   ```
   #mount /dev/hda1 /installpartition
   ```

1. Legen Sie die Knoppix-CD ein und kopieren Sie als *root* das */KNOPPIX*-Verzeichnis der CD in das root-Verzeichnis der Install-Partition. Dabei ist zu beachten, dass die Rechte erhalten bleiben:
   `#cp -a /mnt/cdrom/KNOPPIX /installpartition/`

2. Folgende Dateien von der CD müssen ebenfalls in die root-Partition kopiert werden:
   autorun.bat, cdrom.ico, autorun.inf, index.html

Die Verzeichnisstruktur sollte jetzt so aussehen:

Schritt 3: Bootimage kopieren

Das Bootimage enthält den komprimierten Kernel, die Initial Ramdisk und den Bootloader. Um den Inhalt des Images */KNOPPIX/boot.img* auf Festplatte zu kopieren, muss es Loop-gemounted werden:

```
#mount /mnt/cdrom/KNOPPIX/boot.img /mnt/floppy -o loop
#cp /mnt/floppy/* /mnt/installpartition/harddrive_boot
```

Schritt 4: Partition bootbar machen

Um die Partition bootbar zu machen, muss der Bootloader angepasst werden. Hier ein Beispiel mit LILO:

```
image=/mnt/installpartition/harddrive_boot/linux26
label="knoppix"
initrd=/mnt/installpartition/harddrive_boot/minirt26.gz
read-only root=/dev/hda9
```

Nach dem Ausführen von *lilo -v* ist ein Booten vom installierten Knoppix möglich.

3.5.3 Vorbereiten der Entwicklungsumgebung

Knoppix ist nun auf Festplatte installiert und die Entwicklungsumgebung kann erstellt werden. Die Entwicklungsumgebung beinhaltet:

- Eine lauffähige Kopie des Dateisystems, also das, was Sie entwickeln werden.
- Eine Kopie des Originalsystems für den Fall, dass Sie mit dem Ergebnis des Remastering nicht zufrieden sind und neu anfangen wollen.
- Skripte und Tools, die nötig sind, das neue Live-CD-Filesystem und das ISO-Image zu erzeugen.
- ggf. eine Swap-Datei, falls Sie weniger als 1GB RAM besitzen.

Schritt 1: Booten des installierten Knoppix

Sie werden später eine lauffähige Kopie des Dateisystems auf Festplatte kopieren. Dieser Vorgang benötigt mindestens 3GB freien Festplattenplatz und wird auf der Entwicklungspartition vorgenommen. Alle folgenden Schritte werden falls nicht anders angegeben auf der Entwicklungspartition ausgeführt.

Schritt 2: Verzeichnisse anlegen

- Legen Sie auf der Entwicklungspartition folgende Verzeichnisse an: *KNOPPIX_original, KNOPPIX_remastered* und *masterISO.*
- Gehen Sie ins *masterISO*-Verzeichnis und legen Sie symbolische Verknüpfungen an, die auf die Verzeichnisstruktur der Knoppix-HD-Partition zeigen:
  KNOPPIX -> /installpartition/KNOPPIX
  boot -> /installpartition/harddrive_boot
  Ausserdem kopieren Sie die autorun.bat, autorun.inf, cdrom.ico und index.html in das *masterISO*-Verzeichnis.

Schritt 3: Kopieren des Knoppix Filesystems

Kopieren Sie den Inhalt des KNOPPIX-Verzeichnisses von der CD auf die Festplatte. Dabei ist es wichtig zu beachten, dass die Dateiattribute erhalten bleiben müssen! Das Kopieren dauert eine Weile...

```
#cp -pR /KNOPPIX/* /entwicklungspartition/KNOPPIX_original
#cp -pR /entwicklungspartition/KNOPPIX_original /entwicklungspartition/KNOPPIX_remastered
```

Schritt 4: Anlegen der Skripte

Da im Laufe der Entwicklung einige Vorgänge immer wieder vorkommen, empfiehlt es sich, Skripte anzulegen, die einem die Arbeit erleichtern und Zeit sparen.

Zuerst sollte der Swap-Speicher angelegt werden, falls nicht genug Arbeitsspeicher zur Verfügung steht. *1.createSwap.sh* enthält die nötigen Anweisungen:

```
#!/bin/sh
# 1.createSwap.sh
# funktioniert nur, wenn vorher das swapfile angelegt
# wurde, z.B. mit:
# dd if=/dev/zero of=swapfile bs=1024 count=1000000 (Bsp:1GB)

/sbin/mkswap ./swapfile
/sbin/swapon ./swapfile
```

Um sicherzustellen, dass das Anlegen und aktivieren funktioniert hat, reicht ein Aufruf von ***free -otm*** aus.

2.testDistribution.sh erstellt ein komprimiertes Filesystem mit Hilfe von *mkisofs* und dem Utility *create_compressed_fs* von der Knoppix-CD. Dazu muss *create_compressed_fs* in das momentane Arbeitsverzeichnis kopiert werden. Hier der Inhalt des Skripts:

```
#!/bin/sh
# 2.testDistribution.sh
mkisofs -R -U -hide-rr-moved -cache-inodes -no-bak -pad \
./KNOPPIX_remastered | nice -5 ./create_compressed_fs \
> ./masterISO/KNOPPIX/KNOPPIX
```

Erklärung der Optionen:

- **-R:** das Rock Ridge-Protokoll wird benutzt um die Dateien auf dem iso9660-Filesystem zu beschreiben
- **-U:** lässt Dateinamen zu, die den iso9660-Standard nicht einhalten
- **-hide-rr-moved:** benennt das RR_MOVED-Verzeichnis in das versteckte Verzeichnis *.rr_moved* um. In diesem Verzeichnis befinden sich der Verzeichnisbaum, er wird für den herkömmlichen Nutzer versteckt.
- **-cache-inodes:** bewirkt, dass Hardlinks gefunden werden und eine Datei mit mehreren Namen nur einmal auf der CD vorkommt, was Platz spart
- **-no-bak:** Backup-Dateien, die ~ oder # enthalten bzw. auf *.bak* enden, werden nicht miteinbezogen
- **-pad:** falls die Gesamtgrösse kein Vielfaches von 16Bit ist, wird solange aufgefüllt, bis ein Vielfaches erreicht ist

Nach dem Ausführen dieses Skripts, was übrigens je nach Rechner gut 30Minuten dauern kann, kann man die gemachten Änderungen testen, indem man das System mit dem installierten Knoppix bootet. Das gestartete System entspricht der nachher auf CD gebrannten Distribution. Zum Brennen muss ein ISO-Image erstellt werden, hier die *3.createLiveCD.sh*:

```
#!/bin/sh
mkisofs -pad -f -l -r -J -v -V "KNOPPIX" \
-b KNOPPIX/boot.img -c KNOPPIX/boot.cat \
-o ./knoppix.iso ./masterISO/
```

Hier die Schaltererklärungen:

- **-f:** symbolische Links werden verfolgt
- **-l:** erlauben von 31 Zeichen langen Dateinamen, statt dem 8.3-Format des iso9660-Standards
- **-r:** Die User-ID und die Group-ID von Dateien und Verzeichnissen wird auf 0 gesetzt; damit werden sie für den Nutzer nicht lesbar. Sinnvoll ist das, weil diese Dateien für den Nutzer unwichtig sind, für den Author der CD allerdings nicht.
- **-J:** zusätzlich werden Joilet-Verzeichniseinträge geschrieben, damit die CD auch unter Windows genutzt werden kann

- **-v:** verifizieren jeder Aktion
- **-V "name":** Volume ID wird auf *name* gesetzt
- **-b dateiname:** spezifiziert den Pfad und Dateinamen, der benutzt wird, falls eine "El Torito"-bootbare CD erstellt wird
- **-c dateiname:** gibt den Namen der Boot-Katalog-Datei an, die beim Bau einer "El Torito"-bootbaren CD benötigt wird

Das fertige ISO-Image kann nun auf CD gebrannt werden!

Der Grundstein für das Remastern ist hiermit gelegt. Es kann nun damit begonnen werden, überflüssige Pakete zu entfernen oder andere hinzuzufügen. Eigene Skripte können erstellt werden und das gesamte System eigenen Vorstellungen angepasst werden.

3.6 Entwicklung

3.6.1 Entfernen/Installieren von Paketen

Als erstes muss ins Verzeichnis */entwicklungspartition/KNOPPIX_remastered* gewechselt und *chroot* ausgeführt werden, um ein neues root-Verzeichnis zu bekommen:

```
#chroot ./
```

Dabei kann es vorkommen, dass Meldungen wie */dev/null permission denied* erscheinen. Falls dies der Fall ist, muss die Partition mit der Option *nodev* neu gemounted werden. Um Geräte oder das Netz nutzen zu können, muss das */proc*-Filesystem ebenfalls eingehängt werden.

```
#mount -t proc /proc proc
```

Beim Verlassen des chroot-Systems auch daran denken, */proc* auszuhängen. Mit dem Tool *apt-get* können Pakete installiert und deinstalliert werden. Als erstes muss jedoch *apt-get update* ausgeführt werden. Dieser Aufruf holt aus dem Netz aktuelle Versionen der Softwarelisten, damit keine veraltete Software installiert wird.

Für das Praktikum wurde das OpenOffice-Paket entfernt, was 200MB mehr Platz brachte. Dafür wurden Tools wie *ngrep, tcpdump* und der Nameserver *bind8* installiert.

Jetzt ist es Zeit für den ersten Test, also wird das proc-FS ausgehängt, die chroot-Umgebung verlassen, das Skript *2.testDistribution.sh* ausgeführt und dann rebootet, um Knoppix von Festplatte zu starten und die Veränderungen zu testen. Man muss also nicht extra nach jeder Veränderung des Systems eine CD brennen um die Änderungen testen zu können.

3.6.2 Bootvorgang erweitern

Der normale Bootvorgang von Knoppix wird dahingehend erweitert, dass aus dem Skript *knoppix.sh* ein weiteres Skript aufgerufen wird, das den User nach der IP-Adresse und Subnetzmaske fragt, die der Router hat. Der Benutzer hat auch die Möglichkeit, verschiedene Dienste, die benötigt werden, starten zu lassen.

Damit die Benutzerinteraktion übersichtlich und einfach bleibt, wir das Tool *dialog* verwendet, das unterschiedliche Dialogboxen grafisch darstellen kann. Es ermöglicht neben Checklisten, Radiolisten und Messageboxen auch Inputboxen, mit deren Hilfe Usereingaben gespeichert und später weiterverarbeitet werden können.

knoppix.sh wird um die Zeile "*sh /etc/init.d/kx_ip_erfragen.sh" (vgl. Abschnitt 3.8.5)* ergänzt und startet somit die Ausführung eines Skripts, das mit Hilfe von *dialog* den Benutzer nach den IP-Adressen und der zugehörigen Subnetzmasken fragt, die der Routerrechner, in dem die CD liegt, erhalten soll. Zuerst wird nach der externen IP/Subnetzmaske gefragt, die dem Interface eth0 zugewiesen wird. Anschliessend nach der internen IP/Subnetzmaske, die dem Netzwerkinterface eth1 zugeordnet wird. Dieses Skript veranlasst den Ablauf eines weiteren Shell-Skripts, das eine Auswahl an Diensten zur Verfügung stellt, die gestartet werden können. Je nach Versuchstag im IT-Sicherheitspraktikum müssen vom Benutzer unterschiedliche Dienste ausgewählt werden. Wird beispielsweise eine Firewall benötigt, reicht das Markieren des entsprechenden Dienstes im Auswahlmenü. Zur Auswahl stehen:

- Firewall (iptables) (vgl. Abschnitt 3.8.2)
- Network Address Translation (NAT) & IP-Spoof-Protection (vgl. Abschnitt 3.8.3)
- VPN (FreeS/WAN, IPsec) (vgl. Abschnitt 3.8.12)
- Nameserver (bind8) (vgl. Abschnitt 3.8.6 & 3.8.7)
- Webserver (apache)
- Mailserver (sendmail) (vgl. Abschnitt 3.8.8)
- Telnet- & SSH-Server (vgl. Abschnitt 3.8.9 & 3.8.13)
- Intrusion Detection System (snort)
- Proxy (squid) (vgl. Abschnitt 3.8.10)
- socks-Proxy (dante) (vgl. Abschnitt 3.8.11)

Zu jedem dieser Punkte existiert ein passendes Skript, das den entsprechenden Dienst konfiguriert und startet. Die zur Konfiguration erforderlichen Daten wie z.B. die IP-Adresse werden dabei aus einer während dem Erfragen erstellten Datei gelesen.

Nachdem die benötigten Shell-Skripte fertig abgearbeitet sind, startet das System wie gewohnt weiter, es wird also auch die grafische Benutzeroberfläche geladen. Alle benötigten Dienste sind nun so konfiguriert, dass die am Partnerrechner arbeitende Gruppe keinerlei Probleme oder Einschränkungen hat, während sie die Versuche bearbeiten.

3.7 Zusammenfassung und Ausblick

Die gesetzten Ziele des Forgeschrittenenpraktikums konnten nur teilweise erreicht werden. Die Erstellung der Live-CD wurde erfolgreich abgeschlossen und ist bereits in Betrieb. Fehlende Gruppen können problemlos ersetzt werden. Damit sind sowohl für die Studenten als auch für die Betreuer verbesserte Arbeitsbedingungen geschaffen worden.

Das Update konnte leider aufgrund der beschriebenen Probleme nicht vollzogen werden. Die Rechner laufen weiterhin mit Suse 8.0 und alle Änderungen wurden rückgängig gemacht. Ein Update mit einer aktuelleren Suse-Version sollte aber problemloser ablaufen, da der Kernel und die betroffenen Programme weiterentwickelt wurden.

3.8 Anhang

3.8.1 *Dienstauswahl (kx_ dienste.sh):*

```
#!/bin/sh
# /etc/init.d/kx_dienste.sh
# Per ‘‘dialog’’ abfragen, welche Dienste gestartet werden sollen?
rm -f /tmp/dienste.txt
dialog --backtitle Auswahl --title Dienste \
  --checklist "Welche Dienste sollen gestartet werden?" 15 60 10 \
  "Firewall" "(iptables)" on \
  "Anti-Spoofing & NAT" "(Versuch 4)" off \
  "VPN" "FreeS/Wan, ipsec)" off \
  "Nameserver" "(named)" off \
  "Proxy" "(squid)" off \
  "Telnet-&SSH-Server" "(telnetd/sshd)" off \
  "Mailserver" "(sendmail)" off \
  "Webserver" "(apache)" off \
  "Socks" "(dante-client)" off \
  "IDS" "(snort)" off \ 2> /tmp/dienste.txt
if [ ‘cat /tmp/dienste.txt | grep -c "Firewall"‘ -eq "1" ]; then
  echo "Starte Firewall"
  /etc/init.d/kx_fw_dyn.sh start
fi
if [ ‘cat /tmp/dienste.txt | grep -c "Anti-Spoofing & NAT"‘ -eq "1" ]; then
  echo "Starte Anti-Spoofing und NAT"
  /etc/init.d/kx_fw_nat.sh start
fi
if [ ‘cat /tmp/dienste.txt | grep -c "VPN"‘ -eq "1" ]; then
  echo "Starte VPN"
  /etc/init.d/kx_vpn.sh start
  /etc/init.d/ipsec stop
  /etc/init.d/ipsec start
fi
if [ ‘cat /tmp/dienste.txt | grep -c "Nameserver"‘ -eq "1" ]; then
  echo "Starte Nameserver"
  /etc/init.d/kx_fw_dyn.sh stop
  /etc/init.d/kx_named.conf.options.sh
  /etc/init.d/kx_named.conf.sh
  /etc/init.d/bind stop
  /etc/init.d/bind start
  /etc/init.d/kx_resolv.conf_named.sh
fi
if [ ‘cat /tmp/dienste.txt | grep -c "Proxy"‘ -eq "1" ]; then
  echo "Starte Proxy (und schalte vorher Firewall aus)"
  /etc/init.d/kx_fw_dyn.sh stop
  /etc/init.d/kx_squid.sh
```

```
  /etc/init.d/squid start
fi
if [ `cat /tmp/dienste.txt | grep -c "Telnet-&SSH-Server"` -eq "1" ]; then
  echo "Starte Telnet- & SSH-Server"
  /etc/init.d/kx_telnetssh.sh
fi
if [ `cat /tmp/dienste.txt | grep -c "Mailserver"` -eq "1" ]; then
  echo "Starte Mailserver"
  /etc/init.d/kx_mail_mailertable.sh
  /etc/init.d/sendmail start
fi
if [ `cat /tmp/dienste.txt | grep -c "Webserver"` -eq "1" ]; then
  echo "Starte Webserver"
  /etc/init.d/apache start
fi
if [ `cat /tmp/dienste.txt | grep -c "Socks"` -eq "1" ]; then
  echo "Konfiguriere Socks-Client"
  /etc/init.d/kx_socks.sh
fi
if [ `cat /tmp/dienste.txt | grep -c "IDS"` -eq "1" ]; then
  echo "Starte Snort"
  /etc/init.d/snort start
fi
```

3.8.2 *Dynamische Firewall (kx_fw_ dyn.sh):*

```
#!/bin/sh
# /etc/init.d/kx_fw_dyn.sh
# Hiermit wird eine dynamische iptables-Firewall abhängig von den
# Usereingaben (kx_ip_erfragen.sh) erstellt.
IPTABLES=/sbin/iptables
case "$1" in
start)
 set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
 IP_INT=$1
 SUB_INT=$2
 set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
 IP_EXT=$1
 SUB_EXT=$2
 set $(echo $IP_INT | tr "." " ")
 IP_INT_LAST=$4
 IP_INT_LAST_P=`expr $4 + 1`
 IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P
 echo "IP des Partnerrechners: $IP_PARTNER"

 rm -f /tmp/fw_fertig.txt

 echo "localhost erlauben"
 $IPTABLES -A INPUT -i lo -j ACCEPT $IPTABLES -A OUTPUT -o lo -j ACCEPT
 echo -n "Ping/Traceroute erlauben..."
 #ping
 $IPTABLES -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
 $IPTABLES -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
 $IPTABLES -A FORWARD -p icmp --icmp-type echo-request -j ACCEPT
 #traceroute
 $IPTABLES -A INPUT -p udp --dport 33000: -j ACCEPT
 $IPTABLES -A OUTPUT -p udp --dport 33000: -j ACCEPT
 $IPTABLES -A FORWARD -p udp --dport 33000: -j ACCEPT
 echo " done"
 echo "Ausgehende DNS-Abfragen"
 $IPTABLES -A OUTPUT -p udp --sport 1024: --dport 53 -m state --state NEW -j ACCEPT
 $IPTABLES -A FORWARD -p udp --sport 1024: --dport 53 -m state --state NEW -j ACCEPT
 echo "Ausgehende Telnet-Verbindungen"
 $IPTABLES -A OUTPUT -p tcp --sport 1024: --dport 23 --syn -j LOG
 $IPTABLES -A OUTPUT -p tcp --sport 1024: --dport 23 -m state --state NEW -j ACCEPT
 $IPTABLES -A FORWARD -p tcp --sport 1024: --dport 23 --syn -j LOG
 $IPTABLES -A FORWARD -p tcp --sport 1024: --dport 23 -m state --state NEW -j ACCEPT
 echo "Ausgehende FTP-Verbindungen"
# FTP-Kommando-Verbindung, Rest automatisch
 $IPTABLES -A OUTPUT -p tcp --sport 1024: --dport 21 --syn -j LOG
 $IPTABLES -A OUTPUT -p tcp --sport 1024: --dport 21 -m state --state NEW -j ACCEPT
 $IPTABLES -A FORWARD -p tcp --sport 1024: --dport 21 --syn -j LOG
```

```
$IPTABLES -A FORWARD -p tcp --sport 1024: --dport 21 -m state --state NEW -j ACCEPT
echo "Ausgehende SSH-Verbindungen"
$IPTABLES -A OUTPUT -p tcp --sport 1024: --dport 22 --syn -j LOG
$IPTABLES -A OUTPUT -p tcp --sport 1024: --dport 22 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 1024: --dport 22 --syn -j LOG
$IPTABLES -A FORWARD -p tcp --sport 1024: --dport 22 -m state --state NEW -j ACCEPT
echo "Ausgehende SMTP-Verbindungen"
$IPTABLES -A OUTPUT -p tcp --sport 1024: --dport 25 --syn -j LOG
$IPTABLES -A OUTPUT -p tcp --sport 1024: --dport 25 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 1024: --dport 25 --syn -j LOG
$IPTABLES -A FORWARD -p tcp --sport 1024: --dport 25 -m state --state NEW -j ACCEPT
echo "Ausgehende Port 3128(Proxy)-Verbindungen"
$IPTABLES -A OUTPUT -p tcp --sport 1024: --dport 3128 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 1024: --dport 3128 -m state --state NEW -j ACCEPT
echo "Eingehendes SSH vom Secserver"
$IPTABLES -A INPUT -p tcp -s 192.168.216.254 --sport 1024: --dport 22 --syn -j LOG
$IPTABLES -A INPUT -p tcp -s 192.168.216.254 --sport 1024: --dport 22 -m state --state
NEW -j ACCEPT
echo "Eingehendes SSH vom Partnerrechner $IP_PARTNER"
$IPTABLES -A INPUT -p tcp -s $IP_PARTNER --sport 1024: --dport 22 --syn -j LOG
$IPTABLES -A INPUT -p tcp -s
$IP_PARTNER --sport 1024: --dport 22 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 1024: --dport 22 --syn -j LOG
$IPTABLES -A FORWARD -p tcp --sport 1024: --dport 22 -m state --state NEW -j ACCEPT
echo "Eingehendes Telnet vom Secserver"
$IPTABLES -A INPUT -p tcp -s 192.168.216.254 --sport 1024: --dport 23 --syn -j LOG
$IPTABLES -A INPUT -p tcp -s 192.168.216.254 --sport 1024: --dport 23 -m state --state
NEW -j ACCEPT
echo "Eingehendes Telnet vom Partnerrechner $IP_PARTNER"
$IPTABLES -A INPUT -p tcp -s
$IP_PARTNER --sport 1024: --dport 23 --syn -j LOG
$IPTABLES -A INPUT -p tcp -s $IP_PARTNER --sport 1024: --dport 23 -m state --state
NEW -j ACCEPT
$IPTABLES -A FORWARD -p tcp --sport 1024: --dport 23 --syn -j LOG
$IPTABLES -A FORWARD -p tcp --sport 1024: --dport 23 -m state --state NEW -j ACCEPT
echo "Eingehendes FTP"
$IPTABLES -A INPUT -p tcp --sport 1024: --dport 21 --syn -j LOG
$IPTABLES -A INPUT -p tcp --sport 1024: --dport 21 -m state --state NEW -j ACCEPT
echo "Eingehendes DNS"
$IPTABLES -A INPUT -p udp --sport 1024: --dport 53 -m state --state NEW -j ACCEPT
echo "Eingehendes NTP"
$IPTABLES -A INPUT -p udp --sport 1024: --dport 123 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -p udp --sport 1024: --dport 123 -m state --state NEW -j ACCEPT
echo "Eingehendes SMTP"
$IPTABLES -A INPUT -p tcp --sport 1024: --dport 25 --syn -j LOG
$IPTABLES -A INPUT -p tcp --sport 1024: --dport 25 -m state --state NEW -j ACCEPT
echo "Eingehender Squid"
```

```
 $IPTABLES -A INPUT -p tcp --sport 1024: --dport 3128 -m state --state NEW -j ACCEPT
 $IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
 $IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
 $IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
 $IPTABLES -P INPUT DROP
 $IPTABLES -P OUTPUT DROP
 $IPTABLES -P FORWARD DROP
 echo "1" > /tmp/fw_fertig.txt
 ;;
stop)
 $IPTABLES -F
 $IPTABLES -F -t nat
 $IPTABLES -P INPUT ACCEPT
 $IPTABLES -P OUTPUT ACCEPT
 $IPTABLES -P FORWARD ACCEPT
 echo "Firewall gestoppt"
 ;;
restart)
 $0 stop
 $0 start
 echo "Firewall restartet"
 ;;
*)
 echo "Usage $0 {start|stop|restart}"
 ;;
esac
```

3.8.3 *NAT (kx_fw_nat.sh):*

```
#!/bin/sh
# /etc/init.d/kx_fw_nat.sh
# Network Adress Translation wird aktiviert.
IPTABLES=/sbin/iptables
case "$1" in
 start)
  $IPTABLES -F -t nat
  if ! [ -e /tmp/fw_fertig.txt ];then /
    etc/init.d/kx_fw_dyn.sh start
  fi
  set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
  IP_INT=$1
  SUB_INT=$2
  set $(echo $IP_INT | tr "." " ")
  IP_INT_LAST=$4
  IP_INT_LAST_P=`expr $4 + 1`
  IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P
  if [ $IP_INT_LAST = 1 ]; then
    IP_PARTNER10=10.0.0.$IP_INT_LAST_P
    IP_INT10=10.0.0.$IP_INT_LAST
    SUB_INT10=255.255.255.0
  elif [ $IP_INT_LAST = 23 ]; then
    IP_PARTNER10=10.0.1.$IP_INT_LAST_P
    IP_INT10=10.0.1.$IP_INT_LAST
    SUB_INT10=255.255.255.0
  elif [ $IP_INT_LAST = 45 ]; then
    IP_PARTNER10=10.0.2.$IP_INT_LAST_P
    IP_INT10=10.0.2.$IP_INT_LAST
    SUB_INT10=255.255.255.0
  elif [ $IP_INT_LAST = 109 ]; then
    IP_PARTNER10=10.0.3.$IP_INT_LAST_P
    IP_INT10=10.0.3.$IP_INT_LAST
    SUB_INT10=255.255.255.0
  elif [ $IP_INT_LAST = 67 ]; then
    IP_PARTNER10=10.0.4.$IP_INT_LAST_P
    IP_INT10=10.0.4.$IP_INT_LAST
    SUB_INT10=255.255.255.0
  fi
  # Route für die NAT-Adressen vom Partnerrechner
  route add -host $IP_PARTNER10 gw $IP_PARTNER
  # NAT-Umsetzung Ziel-IP-Adresse für alle Pakete zum Partnerrechner
  $IPTABLES -d $IP_PARTNER -t nat -A PREROUTING -j DNAT --to-destination $IP_PARTNER10
  # NAT-Umsetzung für eingehende Pakete vom Partnerrechner
  $IPTABLES -s $IP_PARTNER10 -t nat -A POSTROUTING -j SNAT --to-source $IP_PARTNER
  #Schalte Spoof-Protection ein
```

```
  if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
    echo -n "Schalte Anti-Spoofing-Konfiguration ein (rp_filter)..."
    for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
     echo 1 > $f
    done
    echo " done."
  else
    echo "Konnte Anti-Spoofing nicht einschalten -> manuell einschalten bitte."
  fi
  ;;
stop)
  $IPTABLES -F -t nat
  echo "NAT ausgeschaltet"
  ;;
*)
  echo "Usage: kx_fw_nat.sh {start|stop}"
  exit 0
  ;;
esac
```

3.8.4 Hostname setzen (kx_hostname.sh):

```
#!/bin/sh
# /etc/init.d/kx_hostname.sh
set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
IP_INT=$1
SUB_INT=$2

set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
IP_EXT=$1
SUB_EXT=$2

set $(echo $IP_INT | tr "." " ")
IP_INT_LAST=$4
IP_INT_LAST_P=`expr $4 + 1` IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P

if [ $IP_INT_LAST -eq 1 ];then
  hostname pcsec1
  rm -f /etc/hostname /etc/hosts
  echo "pcsec1" > /etc/hostname
  echo "127.0.0.1 localhost" > /etc/hosts
  echo "$IP_INT pcsec1" >> /etc/hosts

elif [ $IP_INT_LAST -eq 23 ];then
  hostname pcsec3
  rm -f /etc/hostname /etc/hosts
  echo "pcsec3" > /etc/hostname
  echo "127.0.0.1 localhost" > /etc/hosts
  echo "$IP_INT pcsec3" >> /etc/hosts

elif [ $IP_INT_LAST -eq 45 ];then
  hostname pcsec5
  rm -f /etc/hostname /etc/hosts
  echo "pcsec5" > /etc/hostname
  echo "127.0.0.1 localhost" > /etc/hosts
  echo "$IP_INT pcsec5" >> /etc/hosts

elif [ $IP_INT_LAST -eq 67 ];then
  hostname pcsec7
  rm -f /etc/hostname /etc/hosts
  echo "pcsec7" > /etc/hostname
  echo "127.0.0.1 localhost" > /etc/hosts
  echo "$IP_INT pcsec7" >> /etc/hosts

elif [ $IP_INT_LAST -eq 109 ];then
  hostname pcsec9
  rm -f /etc/hostname /etc/hosts
  echo "pcsec9" > /etc/hostname
  echo "127.0.0.1 localhost" > /etc/hosts
  echo "$IP_INT pcsec9" >> /etc/hosts
fi
```

3.8.5 Topologie, IP & Subnetzmaske erfragen (kx_ip_erfragen.sh)

```
#!/bin/sh

      # Mit diesem Skript wird die IP-Adresse erfragt, die zur weiteren
      # Systemkonifuration nötig ist. Auf ihr bauen verschiedene
      # Konfigurationsdateien auf (squid, named, firewall...).
      # Ausserdem wird erfragt, welche Dienste gestartet werden sollen.
      # WELCHE TOPOLOGIE WIRD VERWENDET? rm -f /tmp/topologie.txt
      dialog --backtitle Auswahl --title Topologie \
        --radiolist "Hub- oder Sterntopologie?" 10 60 2 \
        "Hubtopologie" "(Versuch 1 & 2)" off \
        "Sterntopologie" "(restliche Versuche)" on 2> /tmp/topologie.txt
      topo=$(cat /tmp/topologie.txt)
      rm -f /tmp/ipsub_extern.txt /tmp/ipsub_intern.txt
      fertig=1

      while [ $fertig -eq 1 ]; do
        fertig=0
        dialog --title IP-Adresse \
        --inputbox "Gib die EXTERNE IP-Adresse (eth0) mit Subnetzmaske \
      dieses Rechners ein (zB: 192.168.216.205/255.255.255.192)" 10 60 2> /tmp/ipsub_extern.txt

      set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
      IP_EXT=$1
      SUB_EXT=$2

      if [ -z $SUB_EXT ]; then
        dialog --msgbox "Keine Subnetzmaske angegeben" 10 60 fertig=1
      fi
      done

      fertig=1
      while [ $fertig -eq "1" ]; do
        fertig=0
        dialog --title IP-Adresse \
          --inputbox "Gibt die INTERNE IP-Adresse (eth1) mit Subnetzmaske \
      dieses Rechners ein (zB: 192.168.216.45/255.255.255.240)" 10 60 2> /tmp/ipsub_intern.txt


      set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
      IP_INT=$1
      SUB_INT=$2

      if [ -z $SUB_INT ]; then
        dialog --msgbox "Keine Subnetzmaske angegeben" 10 60 fertig=1
      fi
      done
```

3.8.6 Name-Server-Optionen (kx_named.conf.options.sh):

```
#!/bin/sh
#erstellt /etc/bind/named.conf.options

set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
IP_INT=$1
SUB_INT=$2

set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
IP_EXT=$1
SUB_EXT=$2

set $(echo $IP_INT | tr "." " ")
IP_INT_LAST=$4
IP_INT_LAST_P=`expr $4 + 1` IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P

rm -f /etc/bind/named.conf.options
cp /KNOPPIX/etc/bind/named.conf.options /etc/bind/


echo "options {" > /etc/bind/named.conf.options
echo " directory \"/var/cache/bind\";" >> /etc/bind/named.conf.options

echo " fetch-glue no;" >> /etc/bind/named.conf.options
echo " forwarders {" >> /etc/bind/named.conf.options
echo " 192.168.216.254;" >> /etc/bind/named.conf.options
echo " };" >> /etc/bind/named.conf.options
echo " listen-on {" >> /etc/bind/named.conf.options
echo " 127.0.0.1;" >> /etc/bind/named.conf.options
echo " $IP_EXT;" >> /etc/bind/named.conf.options
echo " };" >> /etc/bind/named.conf.options
echo " allow-transfer {" >> /etc/bind/named.conf.options
echo " 127.0.0.1;" >> /etc/bind/named.conf.options
echo " 192.168.216.254;" >> /etc/bind/named.conf.options
echo " $IP_EXT;" >> /etc/bind/named.conf.options
echo " $IP_PARTNER;" >> /etc/bind/named.conf.options
echo " };" >> /etc/bind/named.conf.options
echo " allow-query {" >> /etc/bind/named.conf.options
echo " 127.0.0.1;" >> /etc/bind/named.conf.options
echo " 192.168.216.254;" >> /etc/bind/named.conf.options
echo " $IP_EXT;" >> /etc/bind/named.conf.options
echo " };" >> /etc/bind/named.conf.options
echo "};" >> /etc/bind/named.conf.options
```

3.8.7 Name-Server Teil 2 (kx_name.conf.sh):

```
#!/bin/sh

# Erstellt die bind8-Config-Datei /etc/bind/named.conf

set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
IP_INT=$1
SUB_INT=$2

set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
IP_EXT=$1
SUB_EXT=$2

set $(echo $IP_INT | tr "." " ")
IP_INT_LAST=$4
IP_INT_LAST_P=`expr $4 + 1` IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P

rm -f /etc/bind/named.conf cp /KNOPPIX/etc/bind/named.conf /etc/bind/

echo "include \"/etc/bind/named.conf.options\";" > /etc/bind/named.conf

echo "logging {" >> /etc/bind/named.conf
echo " category lame-servers { null; };" >> /etc/bind/named.conf
echo " category cname { null; };" >> /etc/bind/named.conf
echo "};" >> /etc/bind/named.conf
echo "zone \".\" {" >> /etc/bind/named.conf echo " type hint;" >> /etc/bind/named.conf

echo " file \"/etc/bind/db.root\";" >> /etc/bind/named.conf
echo "};" >> /etc/bind/named.conf
echo "zone \"localhost\" {" >> /etc/bind/named.conf
echo " type master;" >> /etc/bind/named.conf
echo " file \"/etc/bind/db.local\";" >> /etc/bind/named.conf
echo "};" >> /etc/bind/named.conf
echo "zone \"127.in-addr.arpa\" {" >> /etc/bind/named.conf
echo " type master;" >> /etc/bind/named.conf
echo " file \"/etc/bind/db.127\";" >> /etc/bind/named.conf
echo "};" >> /etc/bind/named.conf
echo "zone \"0.in-addr.arpa\" {" >> /etc/bind/named.conf
echo " type master;" >> /etc/bind/named.conf
echo " file \"/etc/bind/db.0\";" >> /etc/bind/named.conf
echo "};" >> /etc/bind/named.conf
echo "zone \"255.in-addr.arpa\" {" >> /etc/bind/named.conf
echo " type master;" >> /etc/bind/named.conf
echo " file \"/etc/bind/db.255\";" >> /etc/bind/named.conf
echo "};" >> /etc/bind/named.conf
echo "zone \"secp.nm.informatik.uni-muenchen.de\" {" >> /etc/bind/named.conf

echo " type slave;" >> /etc/bind/named.conf
echo " file \"/etc/bind/sec-db.secp.nm.informatik.uni-muenchen.de\";" >>
/etc/bind/named.conf
echo " masters {" >> /etc/bind/named.conf
echo " 192.168.216.24;" >> /etc/bind/named.conf
echo " };" >> /etc/bind/named.conf
echo "};" >> /etc/bind/named.conf
echo "zone \"uni-muenchen.de\" {" >> /etc/bind/named.conf
echo " type forward;" >> /etc/bind/named.conf
```

```
echo " forward only;" >> /etc/bind/named.conf
echo " forwarders {" >> /etc/bind/named.conf
echo " 192.168.216.253;" >> /etc/bind/named.conf
echo " };" >> /etc/bind/named.conf
echo "};" >> /etc/bind/named.conf
echo "zone \"216.168.192.in-addr.arpa\" {" >> /etc/bind/named.conf echo
" type slave;" >> /etc/bind/named.conf
echo " file \"/etc/bind/sec-db.216.168.192\";" >> /etc/bind/named.conf
echo " masters {" >> /etc/bind/named.conf
echo " 192.168.216.110;" >> /etc/bind/named.conf
echo " };" >> /etc/bind/named.conf
echo "};" >> /etc/bind/named.conf echo "// add local zone definitions here"
>> /etc/bind/named.conf
echo "include \"/etc/bind/named.conf.local\";" >> /etc/bind/named.conf
```

3.8.8 Mailserver-Konfiguration (kx_mail_mailertable.sh)

```
#!/bin/sh
# erstellt die /etc/mail/mailertable für sendmail
set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
IP_INT=$1
SUB_INT=$2

set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
IP_EXT=$1
SUB_EXT=$2

set $(echo $IP_INT | tr "." " ")
IP_INT_LAST=$4
IP_INT_LAST_P=`expr $4 + 1` IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P

rm -f /etc/mail/mailertable cp /KNOPPIX/etc/mail/mailertable /etc/mail/

rm -f /etc/mail/access /etc/mail/relay-domains

echo "Erstelle /etc/mail/mailertable"
echo "pcsec1.nm.informatik.uni-muenchen.de smtp:192.168.216.201" > /etc/mail/mailertable

echo "pcsec2.nm.informatik.uni-muenchen.de smtp:192.168.216.2" >> /etc/mail/mailertable

echo "pcsec3.nm.informatik.uni-muenchen.de smtp:192.168.216.203" >> /etc/mail/mailertable

echo "pcsec4.nm.informatik.uni-muenchen.de smtp:192.168.216.24" >> /etc/mail/mailertable

echo "pcsec5.nm.informatik.uni-muenchen.de smtp:192.168.216.205" >> /etc/mail/mailertable

echo "pcsec6.nm.informatik.uni-muenchen.de smtp:192.168.216.46" >> /etc/mail/mailertable

echo "pcsec7.nm.informatik.uni-muenchen.de smtp:192.168.216.207" >> /etc/mail/mailertable

echo "pcsec8.nm.informatik.uni-muenchen.de smtp:192.168.216.68" >> /etc/mail/mailertable

echo "pcsec9.nm.informatik.uni-muenchen.de smtp:192.168.216.209" >> /etc/mail/mailertable

echo "pcsec10.nm.informatik.uni-muenchen.de smtp:192.168.216.110" >> /etc/mail/mailertable
echo "Erstelle /etc/mail/access"
echo "$IP_PARTNER RELAY" > /etc/mail/access
echo "secserv.nm.informatik.uni-muenchen.de 550 Relay denied" >> /etc/mail/access

name=$(hostname)
echo $name".nm.informatik.uni-muenchen.de" > /etc/mail/relay-domains
```

3.8.9 Telnet/SSH (kx_telnetssh.sh):

```
#!/bin/sh

set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
IP_INT=$1
```

```
SUB_INT=$2

set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
IP_EXT=$1
SUB_EXT=$2

set $(echo $IP_INT | tr "." " ")
IP_INT_LAST=$4
IP_INT_LAST_P=`expr $4 + 1` IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P

rm -f /etc/hosts.allow /etc/hosts.deny /etc/inetd.conf
cp /KNOPPIX/etc/inetd.conf /etc/
rm -f /etc/ssh/sshd_config
cp /KNOPPIX/etc/ssh/sshd_config /etc/ssh/
rm -f /etc/inetd.conf
cp /KNOPPIX/etc/inetd.conf /etc/

echo "in.telnetd: $IP_PARTNER,192.168.216.254" >> /etc/hosts.allow
echo "sshd: $IP_PARTNER,192.168.216.254" >> /etc/hosts.allow
echo "ALL:ALL" >> /etc/hosts.deny
echo "ListenAddress $IP_EXT" >> /etc/ssh/sshd_config
echo "sshd: $IP_PARTNER,192.168.216.254" >> /etc/inetd.conf
echo "telnet stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.telnetd"
>> /etc/inetd.conf

/etc/init.d/inetd restart
echo "Telnet und SSH gestartet"
```

3.8.10 Squid-Proxy (kx_squid.sh):

```
#!/bin/sh
# Erstellt die Squid-Config /etc/squid.conf
#Trage Secserver als einzigen DNS in resolv.conf ein

rm -f /etc/resolv.conf /etc/squid.conf
echo "nameserver 192.168.216.254" > /etc/resolv.conf
echo "domain secp.nm.informatik.uni-muenchen.de" >> /etc/resolv.conf

set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
IP_INT=$1
SUB_INT=$2

set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
IP_EXT=$1
SUB_EXT=$2

set $(echo $IP_INT | tr "." " ")
IP_INT_LAST=$4
IP_INT_LAST_P=`expr $4 + 1` IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P

if [ $IP_INT_LAST -eq 1 ];
then HOST=pcsec1 HOST_P=pcsec2
elif [ $IP_INT_LAST -eq 23 ];
then HOST=pcsec3 HOST_P=pcsec4
elif [ $IP_INT_LAST -eq 45 ];
```

```
then HOST=pcsec5 HOST_P=pcsec6
elif [ $IP_INT_LAST -eq 67 ];
then HOST=pcsec7 HOST_P=pcsec8
elif [ $IP_INT_LAST -eq 109 ];
then HOST=pcsec9 HOST_P=pcsec10
fi

echo "http_port 8888" > /etc/squid.conf
echo "icp_port 3130" >> /etc/squid.conf
echo "cache_log /tmp/squid.log" >> /etc/squid.conf
echo "debug_options ALL,2" >> /etc/squid.conf
echo "cache_peer 192.168.216.254 parent 3128 3130 proxy-only no-query default"
>> /etc/squid.conf
echo "hierarchy_stoplist cgi-bin ?" >> /etc/squid.conf
echo "acl QUERY urlpath_regex cgi-bin \?" >> /etc/squid.conf
echo "no_cache deny QUERY " >> /etc/squid.conf
echo "acl all src 0.0.0.0/0.0.0.0" >> /etc/squid.conf
echo "acl manager proto cache_object" >> /etc/squid.conf
echo "acl localhost src 127.0.0.1/255.255.255.255" >> /etc/squid.conf
echo "acl SSL_ports port 443 563" >> /etc/squid.conf
echo "acl Safe_ports port 80 # http" >> /etc/squid.conf
echo "acl Safe_ports port 21 # ftp" >> /etc/squid.conf
echo "acl Safe_ports port 443 563 # https, snews" >> /etc/squid.conf
echo "acl Safe_ports port 70 # gopher" >> /etc/squid.conf
echo "acl Safe_ports port 210 # wais" >> /etc/squid.conf
echo "acl Safe_ports port 1025-65535 # unregistered ports" >> /etc/squid.conf

echo "acl Safe_ports port 280 # http-mgmt" >> /etc/squid.conf
echo "acl Safe_ports port 488 # gss-http" >> /etc/squid.conf
echo "acl Safe_ports port 591 # filemaker" >> /etc/squid.conf
echo "acl Safe_ports port 631 # cups" >> /etc/squid.conf
echo "acl Safe_ports port 777 # multiling http" >> /etc/squid.conf echo
"acl Safe_ports port 901 # SWAT" >> /etc/squid.conf
echo "acl purge method PURGE" >> /etc/squid.conf
echo "acl CONNECT method CONNECT" >> /etc/squid.conf
echo "acl direkt dstdomain $HOST_P.secp.nm.informatik.uni-muenchen.de \\

$HOST.secp.nm.informatik.uni-muenchen.de $HOST-switch.secp.nm.informatik.uni-muenchen.de"
>> /etc/squid.conf
echo "acl erlaubte-domains dstdomain .de .org" >> /etc/squid.conf
echo "acl intern src $IP_PARTNER $IP_INT $IP_EXT" >> /etc/squid.conf
echo "acl intranet dstdomain .secp.nm.informatik.uni-muenchen.de" >> /etc/squid.conf

echo "acl lokal src 192.168.216.192/255.255.255.192" >> /etc/squid.conf

echo " " >> /etc/squid.conf
echo "http_access allow manager localhost" >> /etc/squid.conf
echo "http_access deny manager" >> /etc/squid.conf
echo "http_access allow purge localhost " >> /etc/squid.conf
echo "http_access deny purge" >> /etc/squid.conf
echo "http_access deny !Safe_ports" >> /etc/squid.conf
echo "http_access deny CONNECT !SSL_ports" >> /etc/squid.conf
echo "http_access allow localhost" >> /etc/squid.conf
echo "http_access allow intern erlaubte-domains" >> /etc/squid.conf echo
"http_access allow lokal direkt" >> /etc/squid.conf
echo "http_access deny all" >> /etc/squid.conf
echo "icp_access allow all" >> /etc/squid.conf
```

```
echo "cache_peer_access 192.168.216.254 allow !direkt" >> /etc/squid.conf

echo "cache_peer_access 192.168.216.254 allow all" >> /etc/squid.conf
echo "cache_mgr root@localhost" >> /etc/squid.conf
echo "visible_hostname $HOST" >> /etc/squid.conf
echo "always_direct allow direkt" >> /etc/squid.conf
echo "never_direct deny direkt" >> /etc/squid.conf
echo "never_direct allow intranet" >> /etc/squid.conf
echo "always_direct deny all" >> /etc/squid.conf echo "never_direct allow
all" >> /etc/squid.conf
```

3.8.11 Socks (kx_socks.sh):

```
#!/bin/sh
# Erstellt die Socks-Client-Konfigurationsdatei /etc/dante.conf

set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
IP_INT=$1
SUB_INT=$2

set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
IP_EXT=$1
SUB_EXT=$2

set $(echo $IP_INT | tr "." " ")
IP_INT_LAST=$4
IP_INT_LAST_P=‘expr $4 + 1‘ IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P

rm -f /etc/dante.conf
cp /KNOPPIX/etc/dante.conf /etc/

echo "resolveprotocol: udp # default" > /etc/dante.conf
echo "route {" >> /etc/dante.conf
echo " from: 0.0.0.0/0 to: 0.0.0.0/0 via: 53.146.187.55 port = 1080" >>
/etc/dante.conf
echo " command: bind" >> /etc/dante.conf
echo "}" >> /etc/dante.conf
echo "route {" >> /etc/dante.conf
echo " from: 0.0.0.0/0 to: $IP_PARTNER/32 via: direct" >> /etc/dante.conf
echo "}" >> /etc/dante.conf
echo "route {" >> /etc/dante.conf
echo " from: 0.0.0.0/0 to: 127.0.0.0/8 via: direct" >> /etc/dante.conf

echo " command: connect udpassociate # everything but bind, bind confuses
us." >> /etc/dante.conf
echo "}" >> /etc/dante.conf
echo "route {" >> /etc/dante.conf
echo " from: 0.0.0.0/0 to: $IP_PARTNER/32 via: direct" >> /etc/dante.conf

echo " command: connect udpassociate # everything but bind, bind confuses
us." >> /etc/dante.conf
echo "}" >> /etc/dante.conf
echo "route {" >> /etc/dante.conf
echo " from: 0.0.0.0/0 to: 0.0.0.0/0 via: $IP_PARTNER port = 1080" >> /etc/dante.conf

echo " protocol: tcp udp # server supports tcp and udp." >> /etc/dante.conf

echo " proxyprotocol: socks_v5 # server supports socks v5." >> /etc/dante.conf

echo " method: none #username # we are willing to authenticate via" >>
/etc/dante.conf
echo " # method none, not username." >> /etc/dante.conf echo "}" >> /etc/dante.conf

echo "route {" >> /etc/dante.conf
echo " from: 0.0.0.0/0 to: . via: $IP_PARTNER port = 1080" >> /etc/dante.conf

echo " protocol: tcp udp # server supports tcp and udp." >> /etc/dante.conf
```

```
echo " proxyprotocol: socks_v5 # server supports socks v5." >> /etc/dante.conf

echo " method: none #username # we are willing to authenticate via" >>
/etc/dante.conf
echo " # method none, not username." >> /etc/dante.conf echo "}" >> /etc/dante.conf
```

3.8.12 VPN/IPsec (kx_vpn.sh):

```
#!/bin/sh

set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
IP_INT=$1
SUB_INT=$2

set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
IP_EXT=$1
SUB_EXT=$2

set $(echo $IP_INT | tr "." " ")
IP_INT_LAST=$4
IP_INT_LAST_P=`expr $4 + 1` IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P

rm -f /etc/ipsec.conf
cp /KNOPPIX/etc/ipsec.conf_VORLAGE /etc/ipsec.conf

if [ $IP_INT_LAST -eq 1 ]; then leftid=pcsec2 rightid=pcsec1
elif [ $IP_INT_LAST -eq 23 ]; then leftid=pcsec4 rightid=pcsec3
elif [ $IP_INT_LAST -eq 45 ]; then leftid=pcsec6 rightid=pcsec5
elif [ $IP_INT_LAST -eq 207 ]; then leftid=pcsec8 rightid=pcsec7
elif [ $IP_INT_LAST -eq 209 ]; then leftid=pcsec10 rightid=pcsec9
fi
echo "conn pcsec$IP_INT_LAST-pcsec$IP_INT_LAST_P" >> /etc/ipsec.conf
echo "leftid=@$leftid" >> /etc/ipsec.conf
echo "left=$IP_PARTNER" >> /etc/ipsec.conf
echo "rightid=@$rightid" >> /etc/ipsec.conf
echo "right=$IP_INT" >> /etc/ipsec.conf
```

3.8.13 SSH-Config (kx_ssh.sh):

```
#!/bin/sh
# Versuch 6: sshd

set $(cat /tmp/ipsub_intern.txt | tr "/" " ")
IP_INT=$1
SUB_INT=$2

set $(cat /tmp/ipsub_extern.txt | tr "/" " ")
IP_EXT=$1
SUB_EXT=$2

set $(echo $IP_INT | tr "." " ")
IP_INT_LAST=$4
IP_INT_LAST_P=`expr $4 + 1` IP_PARTNER=$1.$2.$3.$IP_INT_LAST_P

rm -f /etc/ssh/sshd_config
cp /KNOPPIX/etc/ssh/sshd_config /etc/ssh/
rm -f /etc/inetd.conf
cp /KNOPPIX/etc/inetd.conf /etc/

echo "ListenAddress $IP_EXT" >> /etc/ssh/sshd_config
echo "sshd: $IP_PARTNER,192.168.216.254" >> /etc/inetd.conf
```

4 Quellen

- [1] http://www.nm.ifi.lmu.de/Praktika/secp.shtml
- [2] http://www.suse.de
- [3] Klaus Knopper's Knoppix: www.knopper.net/knoppix
- [4] Remastering Howto: http://www.knoppix.net/docs/index.php/KnoppixRemasteringHowtoDeutsc
- [5]Laufwerke verschlüsseln mit Loop-AES: http://www.pl-berichte.de/t_system/loop-aes.html
- [6] Praktikum IT-Sicherheit: http://www.nm.ifi.lmu.de/Praktika/secp.shtml
- [7] El Torito Standard: http://www.phoenix.com
- [8] http://linux.oreillynet.com/linux/2003/11/20/examples/kicklist.txt
- [9] http://www.nm.informatik.uni-muenchen.de/Literatur/MNMPub/Fopras/illi03/PDF-Version/illi03.pdf