

Automatische Installation und Einrichtung von Web-Applikationen und Diensten auf Linux Systemen

Dokumentation des Pflichtprojekts im Studiengang Medieninformatik – Bachelor

Von Alexander Breunig (ab112@hdm-stuttgart.de)
Simon Zehnder (sz028@hdm-stuttgart.de)

Inhaltsverzeichnis

1. Einführung.....	1
1.1 Motivation.....	1
1.2 Zielsetzung.....	2
2. FAI – Fully Automatic Installation.....	3
2.1 Grundlagen.....	3
2.2 Konzept.....	4
2.2.1 Installationsumgebung.....	4
2.3 Durchführung.....	5
2.3.1 Grundinstallation FAI.....	5
2.3.2 Konfiguration der Serverumgebung.....	6
2.3.3 Konfiguration der FAI-Umgebung.....	8
2.3.4 Einrichten des „Configspace“.....	8
2.3.5 Finale Konfiguration und Tests.....	14
2.4 Fazit.....	15
2.4.1 Security Aspekte.....	15
2.4.2 Mögliche Einsatzbereiche.....	15
2.4.3 Allgemeines Fazit.....	15
3. Richtlinien zur Script-Entwicklung.....	16
3.1. Programmiersprache Python.....	16
3.2 Aufbau der Scripts.....	16
3.3 Grundlegende Richtlinien.....	17
3.4 Ablauf der Scriptentwicklung.....	17
4. Pywebback.....	18
4.1 Grundlagen.....	18
4.1.1 Grundlegender Aufbau.....	18
4.2 Implementierung der Module.....	19
4.2.1 Modul „backupdb“.....	19
4.2.2 Modul „configimport“.....	19
4.2.3 Modul „ftp“.....	20
4.2.4 Modul „rotate“.....	20
4.2.5 Modul „tarbackup“.....	21
4.3 Implementierung des Hauptscripts.....	21
4.4 Webrestore.....	22
4.5 Fazit.....	23
4.5.1 Security-Aspekte.....	23
4.5.2 Mögliche Einsatzbereiche.....	24
4.5.3 Allgemeines Fazit.....	24
5. Pywiki_install.....	25
5.1 Grundlagen.....	25
5.1.1 Grundlegender Aufbau.....	25
5.2 Implementierung der Module.....	27
5.2.1 Modul „configimport“.....	27
5.2.2 Modul „dbsetup“.....	27
5.2.3 Modul „Download“.....	27
5.2.4 Modul „LocalSettings“.....	28
5.2.5 Modul „misc_functions“.....	29
5.2.6 Modul „mwikiHash“.....	29
5.3 Implementierung des Hauptscripts.....	29

5.4 Fazit	30
5.4.1 Security-Aspekte.....	30
5.4.2 Mögliche Einsatzbereiche	30
5.4.3 Allgemeines Fazit.....	30
6. Abschluss.....	31
6.1 Erfüllung der Ziele	31
6.1.1 Serverinstallation/Einrichtung.....	31
6.1.2 Installation der Web-Applikationen.....	32
5.2 Abschließendes Fazit	33
7. Anhang.....	34
7.1 Glossar.....	34
7.2 Linkverzeichnis.....	35
7.2.1 FAI.....	35
7.2.2 Scripts.....	35

1. Einführung

1.1 Motivation

Jeder Informatiker kennt es und hat es wahrscheinlich schon viele male gemacht:
Die Installation von Betriebssystemen.

Wenn nicht gerade ein größeres Versionsupdate ansteht ist dies normalerweise ein recht selten durchgeführter Vorgang, was zahlreiche PCs, Notebooks und vor allem Server anbelangt innerhalb deren Lebenszeit sogar einmalig.

Auch Zeitaufwand und Schwierigkeit einer Betriebssysteminstallation halten sich im Normalfall im Rahmen. Moderne Gui-basierte Installer ermöglichen es selbst Laien alle gängigen Linux-Distributionen innerhalb kurzer Zeit zu installieren.

Anders jedoch in Bereichen, in denen eine große Anzahl von PCs und Servern zum Einsatz kommt wie z.B. in Unternehmen, Schulen, Hochschulen oder Universitäten.

Ein einfaches Update auf ein neues Major-Release einer Distribution, wie es bei den großen, freien Desktop-Distributionen Fedora, Suse und Ubuntu jedes halbe Jahr ansteht wird normalerweise mit einem einfachen Update Befehl des jeweiligen Paket-Management-Tools erledigt. Wird jedoch, wie im IT-Bereich dank Moores-Law und stetig steigender Anforderungen relativ häufig, neue Hardware angeschafft sind je nach Größe der Anschaffung bis zu mehrere tausend Neuninstallationen nötig.

Natürlich lassen sich diese auch von externen Dienstleistern erledigen, jedoch kann hier schnell ein relativ großer Geldbetrag zusammenkommen, was vor allem öffentliche Institutionen schnell zurückschrecken lässt. Außerdem ist Hardware mit vorinstallierten Linux-Systemen äußerst selten

Also wird hierbei schnell nach eigenen, kostengünstigeren Lösungen gesucht.

Von der vollständig manuellen Installation jedes einzelnen Systems, über vorgefertigte Musterlösungen, wie sie in Schulen häufig zum Einsatz kommen, bis zum einfachen klonen mit Hilfe von Images haben diese Verfahren jedoch, neben je nach Vorgehensweise verschieden großen Nachteilen, zwei gemeinsame Hauptnachteile:

Der Zeitaufwand groß und es müssen praktisch für den gesamten Zeitraum der Installation Mitarbeiter dafür abgestellt werden.

Neben den eigentlichen Installationen werden vor allem in jüngerer Zukunft immer häufiger Web-Applikationen verschiedener Art benötigt.

Web-Content-Management-Systeme, Wikis, Blogs, E-Learning Plattformen und viele mehr bestimmen in Firmen, Schulen und Hochschulen immer mehr den Alltag.

Auch hier müssen Installationen zum großen Teil manuell durchgeführt werden.

Wichtige Administrationsaufgaben wie Sicherungen oder Sicherheitsupdates werden häufig vernachlässigt oder sogar komplett vergessen.

Hier kommt dieses Projekt ins Spiel:

Automatische Installation und Einrichtung von Web-Applikationen und Diensten auf Linux Systemen

1.2 Zielsetzung

Serverinstallation/Einrichtung:

- Einrichten eines Linux-Servers mit FAI (Fully Automatic Installation) zur automatischen Installation von Linux-Systemen mittels PXE-Boot, mit Auswahl der geeigneten Distributionen, Installation und Konfiguration des Servers.
- Erstellen von geeigneten Images für die automatisch zu installierenden Web-Server.
- Programmieren eines Python-Scripts zur automatischen Installation und Einrichtung einer LAMP-Umgebung (unter Verwendung von Paket-Management-Tools der jeweiligen Distribution).

Installation der Web-Applikationen:

- Programmieren mehrerer Python-Scripts zur automatischen Installation und Einrichtung verschiedener Web-Applikationen (z.B. für die fünf mit dem größten Verbreitungsgrad).
- Eventuell auch einzelnes Script für alle Web-Applikationen.
- Erstellen eines anpassbaren Muster-Python-Scripts als Vorgabe für zusätzliche Web-Applikationen die nicht bereits im Projekt enthalten sind.
- Integration von rudimentären Administrations-Aufgaben für die Web-Applikationen durch das jeweilige Script (z.B. automatische Sicherheitsupdates, Erstellen von Cronjobs durch das Script für zeitgesteuerte Sicherung von Files und Datenbanken).
- Anwender legt grundlegende Parameter für die Installation der Webanwendungen in Konfigurationsdateien mit vordefinierten Parametern fest (z.B. Zeitpunkt für Sicherungen, Admin-Passwörter, grundlegende-Konfigurationsparameter der verschiedenen Web-Applikationen).
- Scripts sollen auch unabhängig von der FAI-Lösung auf bereits installierten Servern lauffähig sein (Vollständige Trennung von Serverinstallation und Installation der Web-Applikationen, Anpassung an Paket-Management Tools verschiedener Distributionen).

2. FAI – Fully Automatic Installation

2.1 Grundlagen

Was ist FAI?

- Ein System für die unbeaufsichtigte Linux-Installation
- Installiert und konfiguriert das komplette Betriebssystem und alle zusätzlichen Software Pakete
- Zentralisiertes Konfigurations-Management und Administration
- Skalierbare und flexible Rollout Methode für Linux Migration
- **Linux Deployment in wenigen Minuten**

Warum FAI benutzen?

- Manuell Installation dauert Stunden, FAI nur wenige Minuten
- Wiederkehrende Aufgaben sind langweilig und führen zu Fehlern
- Man benötigt ein Infrastruktur-Management
- Man will Zeit sparen

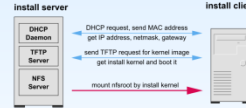
Installationszeiten

CPU + RAM	Software	Zeit
Core i7, 3.2 GHz, 6GB	4.3 GB	7 min
Core i7, 3.2 GHz, 6GB	471 MB	77 s
Core2duo, 2 GHz, 2GB	4.3 GB	17 min
Core2duo, 2 GHz, 2GB	471 MB	165 s
Pentium 4, 3 GHz, 1GB	2200 MB	10 min
Pentium 4, 3 GHz, 1GB	1100 MB	6 min
Pentium 4, 3 GHz, 1GB	300 MB	105 s

Die drei Schritte von FAI

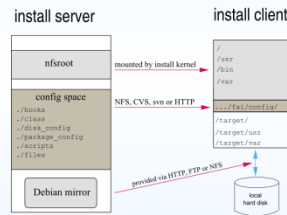
1 - Rechner booten

- Booten via Netzwerk (PXE), CD-ROM oder USB Stick



- Jetzt läuft ein komplettes Linux System unabhängig von der lokalen Festplatte

2 - Konfigurationsdaten holen



3 - Installation durchführen

- Partitionierung der Festplatten und Anlegen von Dateisystemen
- Software-Installation mit dem Paketmanager (apt, yum, yast und andere)
- Konfiguration des OS und der zusätzlichen Programme
- Sichern von Logdateien auf dem Install-Server
- Neustart des fertigen Systems

Voraussetzungen

Ein zu installierender Computer (Install Client):

- Mit Netzwerkkarte
- Mit einer lokalen Festplatte
- Keine Floppy, CD-ROM, Tastatur oder Grafikkarte erforderlich

DHCP Server: Der Install Client erhält Boot- und Konfigurationsdaten von diesem Server

TFTP Server: Von hier lädt der Client seinen Kernel und Start-Konfiguration wenn per PXE über das Netzwerk gestartet wird

Installations-Kernel: Das Installations-System wird mit einem Standard Debian Kernel gestartet

NFS-Root-Dateisystem: Ein Verzeichnis, welches das komplette Installations-System enthält, das die Installation abarbeitet. Alle Install Clients nutzen das gleiche Verzeichnis

Konfigurationsdaten (Configspace): Ein Verzeichnis, das die Konfigurationsdaten für die Install Clients enthält. Dies sind nur einige einfach formatierte und kleine Textdateien in einer vorgegebenen Verzeichnisstruktur

Ein Debian Mirror: Zugriff zu einem Debian Mirror ist notwendig. Die üblichen von apt unterstützten Protokolle sind nutzbar, wenn das Netzwerk den Zugriff darauf erlaubt.

Diese Dienste können auf dem FAI Server liegen, aber auch auf mehrere Rechner verteilt werden.

Funktionen

- Installiert Debian GNU/Linux, Ubuntu, CentOS SuSe, Scientific Linux Cern,
- Unterstützt Xen, KVM und VServer Virtualisierung
- **Klassen-Konzept** unterstützt heterogene Konfiguration und Hardware
- **Regelmäßige Wartung** durch Aktualisierung laufender Systeme ohne Neu-Installation
- **Zentrales Konfigurations-Archiv** für alle Install Clients
- Fortgeschrittenes **Disaster Recovery** System
- Reproduzierbare Installationen
- **Zentrale und automatische Dokumentation**
- Automatische Hardware-Inventarisierung
- Erweiterbar und anpassbar durch Hooks
- **Volle Remote Kontrolle** über ssh während der Installation
- Unterstützung von *Shell, Perl, expect* und *Cfengine* für Konfigurationsskripte
- FAI läuft auf i386, AMD64, PowerPC, Alpha, SPARC, IA64 und IBM z10 Mainframe
- Schnelle Installation für Beowulf- und HPC-Cluster
- Mit FAI-CD kann man den Installations-Prozess auch unabhängig vom Netzwerk und speziellen Diensten durchführen
- **Graphische Administration von FAI mit Gosa**
 - Gosa stellt eine graphische Oberfläche für FAI zur Verfügung
 - Gosa ist ein in PHP entwickeltes LDAP Frontend
 - der FAI Configspace ist komplett im LDAP gespeichert
 - Der Configspace kann über Gosa verwaltet werden

Verfügbarkeit

- Homepage: <http://fai-project.org>
- Open Source unter der GPL Lizenz
- Detaillierte Dokumentation, Mailing Listen, IRC
- Offizielle Debian Pakete, ISO Images der Demo CD
- Kommerzieller Support ist verfügbar

Einige Nutzer von FAI

- Anonym, Finanzindustrie, 32.000 Hosts
- LVM Versicherung, 10.000 Hosts
- StayFriends, 300+ Hosts
- Stadt München, 6000+ - geplant 12.000 Hosts
- Albert Einstein Institut, 1725 Hosts
- Zivit, 260 Hosts auf zwei IBM z10 EC Mainframe
- Archive.org, 200+ Hosts
- XING AG, 300-400 Hosts
- Opera Software, ~300 Hosts
- Stanford Universität, 450 Hosts
- MIT Computer science research lab, 200 Hosts
- The Welcome Trust Sanger Institute, 540 Hosts
- Deutsches Elektronen-Synchrotron, 273 Hosts
- Mobile.de, ~600 Hosts
- Electricité de France (EDF), 1500 hosts
- Linux Information Systems AG, 1000+ Hosts
- ETH Zurich, systems group, ~300 Hosts
- Umeå Universität, 70 Hosts
- Trinity Centre for High Performance Computing, 356 Opteron, 80 Xeons
- High Performance Computing Center North, HPC2N, zwei Cluster mit insgesamt 310 Hosts
- Weitere siehe <http://fai-project.org/reports/>

FAI

Fully Automatic Installation



debian

Plane Deine Installation, und FAI installiert Deinen Plan.

Kontakt:
Thomas Lange
Institut für Informatik, Universität zu Köln
Pohligstraße 1, D – 50969 Köln, Germany
Email: fai@fai-project.org

2.2 Konzept

2.2.1 Installationsumgebung

Zur Verfügung stehende Hardware:

- 4x Dell-Powerededge 750 mit Pentium 4, 40GB-HDD, 256MB Ram, 2x Gigabit Lan
- 8-Port HP-ProCurve Gigabit Switch

Alle Arbeiten an den Servern wurden im Internet-Security-Labor (im folgenden IS-Labor genannt) durchgeführt. Das Netzwerk des IS-Labors ist über einen eigenen Router, der gleichzeitig als DHCP- und DNS-Server fungiert mit dem restlichen Netz der hdm und somit auch mit dem Internet verbunden.

Konfiguration des Switches:

- VLAN2 – Intern, Port 1-4: Testclients
- VLAN3 – Extern, Port 5-8: Verbindung ins Netz des IS-Labors für Internetzugriff

Installation des zentralen Servers für FAI

Grundlegende Installation von Debian-Linux 6.0 mittels eines Netinstall-Images (nur rudimentäre Umgebung, Rest aus dem Internet) ohne zusätzliche Pakete oder manuelle Anpassungen.

Da die zur Verfügung stehenden Server über keine CD- bzw. DVD-Laufwerke verfügen, musste die Installation via USB-Stick durchgeführt werden. Hierzu wurde über das Tool „Unetbootin“ ein USB-Stick mit dem entsprechenden Debian-Image bootfähig gemacht.

Zu Beginn war es nicht möglich von USB-Stick zu booten (Fehlermeldung: „Gave up waiting for root device.“). Nach stundenlangen Versuchen mit alternativen Methoden einen USB-Stick zur Installation vorzubereiten (z.B. „Linux Live USB-Creator“, „Universal USB-Installer“, <http://www.debian.org/releases/stable/i386/ch04s03.html.de>) und der Verwendung von verschiedenen USB-Sticks konnte eine einfache Lösung gefunden werden.

Im Bios des Servers muss unter "Hard-Disk Drive Sequence" der Parameter "Hard-Disk-emulated USB flash drive", an die erste Stelle gesetzt werden. Offensichtlich war der Server zwar in der Lage von USB-Stick zu booten, die Installationsumgebung jedoch sah das primäre Device aus der „Hard-Disk Drive Sequence“ im Bios als Installationsmedium an.

Danach konnte die Installation des zentralen Servers ohne weitere Vorkommnisse abgeschlossen werden. Der zentrale Server erhielt seinen ursprünglichen Hostnamen mit dem er bereits beschriftet war „VADER1“.

2.3 Durchführung

2.3.1 Grundinstallation FAI

Alle für die Installation von FAI benötigten Pakete sind bereits in den Standard Paket-Repositories von Debian und Ubuntu vorhanden.

Durch Aufruf des Befehls „apt-get install fai-quickstart“ erfolgt die Installation des FAI-Servers:

```
Die folgenden zusätzlichen Pakete werden installiert:
  apt-move cfengine2 debconf-utils debootstrap fai-client fai-doc fai-server
  fai-setup-storage isc-dhcp-server libapt-pkg-perl libcrypt-passwdmd5-perl
  libdigest-sha1-perl liblinux-lvm-perl libparse-recdescent-perl
  libproc-daemon-perl libreadline5 lvm2 mdadm nfs-kernel-server openbsd-inetd
  openssh-server parted syslinux-common tftpd-hpa
Vorgeschlagene Pakete:
  debmirror grub perl-tk cryptsetup jfsutils reiserfsprogs xfsprogs
  isc-dhcp-server-ldap ssh-askpass rssh molly-guard ufw parted-doc
Die folgenden NEUEN Pakete werden installiert:
  apt-move cfengine2 debconf-utils debootstrap fai-client fai-doc
  fai-quickstart fai-server fai-setup-storage isc-dhcp-server libapt-pkg-perl
  libcrypt-passwdmd5-perl libdigest-sha1-perl liblinux-lvm-perl
  libparse-recdescent-perl libproc-daemon-perl libreadline5 lvm2 mdadm
  nfs-kernel-server openbsd-inetd openssh-server parted syslinux-common
  tftpd-hpa
0 aktualisiert, 25 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
Es müssen 7.321 kB an Archiven heruntergeladen werden.
Nach dieser Operation werden 19,1 MB Plattenplatz zusätzlich benutzt.
```

Abbildung 1: Installation der FAI-Pakete

Wie auf Abbildung 4 zu sehen, werden über das Paket „fai-quickstart“ sämtliche Abhängigkeiten des FAI-Servers aufgelöst, aber auch Pakete installiert die für die eigentliche Funktionalität des FAI-Servers nicht zwingend benötigt werden z.B. DHCP- und TFTP-Server für den PXE-Boot (FAI kann den Client auch ohne PXE, z.B. über eine bootbare CD booten).

2.3.2 Konfiguration der Serverumgebung

Für Nutzung von PXE-Boot wird generell ein DHCP-Server benötigt, der dem Client beim Start eine eigene IP-Adresse, sowie die IP-Adresse des Servers von dem aus der PXE-Boot erfolgen soll zuweist.

Folgende Möglichkeiten bieten sich hier an:

1) Zuweisung des PXE-Boot-Servers über bereits vorhandenen DHCP-Server

Diese Methode hat den Vorteil, dass alle Clients die sich in derselben Broadcast-Domäne wie der DHCP-Server befinden auch von dem PXE-Boot-Server booten können.

Allerdings könnte es hier passieren, dass ein Client unbeabsichtigt über PXE gebootet wird und somit durch eine fertig eingerichtete FAI-Lösung dessen Betriebssystem komplett überschrieben wird.

Dieses Problem könnte man umgehen, indem man die PXE-Boot-Option in der Konfiguration des DHCP-Servers nur spezifischen MAC-Adressen zuweist. Dies würde jedoch, sofern wie beabsichtigt eine große Anzahl von Clients vom PXE-Boot-Server bedient werden soll, wieder zu einem hohen manuellen Konfigurationsaufwand führen. Außerdem wäre es sinnvoll die frisch installierten Clients vor dem Produktiveinsatz zumindest stichprobenartig zu testen, es können ja z.B. Sicherheitslücken durch manuelle erstellte Scripts und Konfigurationen auftreten.

Daher haben wir uns für die nachfolgende Lösung entschieden:

2) Einrichtung des DHCP-Dienstes auf PXE-Boot-Server

Um die zuvor beschriebenen Nachteile zu umgehen wurde zuerst ein getrenntes VLAN („Intern“) auf den Ports 1-4 des Switches eingerichtet. Hier wurden die zu installierenden Clients und die sekundäre Netzwerkschnittstelle („eth1“) des zentralen Servers angeschlossen werden.

Die primäre Netzwerkschnittstelle („eth0“) des zentralen Servers wurde an Port 5 des Switches angeschlossen. Hier erhält er via DHCP eine Netzwerk-Konfiguration aus dem IS-Labor, welche unter anderem den Internetzugriff ermöglicht. Der Sekundäre Netzwerkadapter erhält die private IP-Adresse 192.168.0.1.

Nachfolgend die Konfiguration der beiden Interfaces des zentralen Servers:

```
# The primary network interface
allow-hotplug eth0
iface eth0 inet dhcp
post-up /etc/network/if-up.d/iptables.sh

# Secondary network interface (intern)
auto eth1
iface eth1 inet static
address 192.168.0.1
netmask 255.255.255.0
pre-up /sbin/ifconfig eth1 up
```

Abbildung 2: Auszug aus /etc/network/interfaces

Der DHCP-Server wurde darauf konfiguriert, nur auf Anfragen die aus dem internen VLAN kommen also an eth1 zu reagieren:

```
# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="eth1"
```

Abbildung 3: Auszug aus /etc/default/isc-dhcp-server

Das interne VLAN soll vom DHCP-Server IP-Adressen aus dem privaten Netz 192.168.0.0/24 erhalten. Während der Installation benötigen die Clients Zugriff auf ein Paket-Repository der jeweils zu installierenden Distribution. Hierzu bietet es sich bei der Installation zahlreicher Clients an, einen lokalen Mirror des Paket-Repositories anzulegen um unnötigen Traffic ins Internet zu vermeiden. In diesem Projekt sollen jedoch die Standard-Paket-Repositories via Internet verwendet werden, da im Rahmen dieses Projektes nur eine geringe Anzahl an Clients für Testzwecke installiert werden muss und die hdm eine schnelle Internetanbindung besitzt.

Hierfür benötigen die Clients aus dem internen VLAN heraus Zugriff ins Internet, also in diesem speziellen Fall in das externe VLAN des Switches.

Für diesen Zweck wurde auf dem zentralen Server das IP-Forwarding aktiviert („net.ipv4.ip_forward = 1“ in „/etc/sysctl.conf“), sowie durch folgenden Shell-Befehl mit Hilfe der iptables Firewall ein NAT-Routing eingerichtet:

```
- iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Dieser Befehl ist bei manuellem Aufruf über die Shell nicht persistent, das heißt nach dem nächsten Neustart des Servers funktioniert das NAT Routing nichtmehr.

Daher wurde der Befehl in die Datei „/etc/network/if-up.d/iptables.sh“ eingetragen, die beim Start der primären Netzwerkschnittstelle („eth0“) aufgerufen wird, wie in der Konfiguration der Netzwerkschnittstellen in Abbildung 1 zu sehen ist.

In der Konfiguration des DHCP-Servers wurden neben dem Subnetz 192.168.0.0/24 für das interne VLAN mit eingeschränkter Range (20 IP-Adressen sollten für Testzwecke genügen), noch die IP-Adresse der sekundären Netzwerkschnittstelle des zentralen Servers („eth1“) als Standard-Gateway (option routers), sowie die IP-Adresse des IS-Labor-Routers als DNS-Server eingetragen. Der PXE-Boot wird über die Parameter „allow booting“, „next-server“ (PXE-Boot-Server) und „filename“ (Linux-Bootloader für PXE-Boot) konfiguriert.

```
subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.100 192.168.0.120;
    option routers 192.168.0.1;
    option domain-name-servers 141.62.66.250;
}
```

```
allow booting;
next-server 192.168.0.1;
filename "pxelinux.0";
```

Abbildung 4: Auszug aus /etc/dhcp/dhcpd.conf

Als letzter Schritt in der Konfiguration der eigentlichen Serverumgebung musste noch der TFTP-Server konfiguriert werden:

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp/fai"
TFTP_ADDRESS="192.168.0.1:69"
TFTP_OPTIONS="--secure"
RUN_DAEMON="yes"
```

Abbildung 5: tftp-server Konfiguration

2.3.3 Konfiguration der FAI-Umgebung

Zunächst musste die Datei „make-fai-nfsroot.conf“ konfiguriert werden:

```
FAI_DEBOOTSTRAP_OPTS="--exclude=info"  
NFSROOT_ETC_HOSTS="192.168.0.1 VADER1"  
FAI_DEBOOTSTRAP="squeeze http://cdn.debian.net/debian"
```

Abbildung 6: Auszug aus /etc/fai/make-fai-nfsroot.conf

Mit „NFSROOT_ETC_HOSTS“ wird der NFS-Server konfiguriert von dem aus der Client die Installationsumgebung sowie den Configspace (Konfigurationsdaten) lädt, in diesem Fall also der zentrale Server. Der Parameter „FAI_DEBOOTSTRAP“ bezeichnet den Debian Paket-Mirror, von dem aus der zentrale Server bei Aufruf des entsprechenden Befehls die grundlegende Installationsumgebung lädt. Es kann Wahlweise ein beliebiger externer Paket-Mirror oder ein lokaler Mirror definiert werden.

Durch Aufruf des Befehls „fai-setup -v“ wird die zuvor bereits erwähnte Installationsumgebung vom entsprechenden Paket-Mirror heruntergeladen und in das Verzeichnis „/srv/fai/nfsroot/live/filesystem.dir“ entpackt. Das entsprechende Kernel-Image wird unter „/srv/tftp“ abgelegt. Die Installationsumgebung stellt ein vollständig lauffähiges Linux-System dar welches unmittelbar nach dem Start des Clients via PXE geladen wird und hat bei der Verwendung von Debian 6.0 Squeeze einen Umfang von ca. 443MB. Als Linux-Distribution zum Installieren der Clients, soll die zu diesem Zeitpunkt aktuelle Version von Ubuntu-Server (10.10) verwendet werden. Ubuntu erfreut sich auch auf Serversystemen immer größerer Beliebtheit und die Verwendung einer aktuelleren, weniger erprobten Version anstatt z.B. Ubuntu 10.04 LTS soll die Schwierigkeit etwas erhöhen. Speziell für Testzwecke ist es ebenfalls möglich eine Installation der zuvor erwähnten, rudimentären Debian-Umgebung durchzuführen (siehe <http://fai-project.org/fai-guide/ar01s02.html>).

2.3.4 Einrichten des „Configspace“

Als Configspace wird das zentrale Konfigurationsverzeichnis („/srv/fai/config“) des FAI-Servers bezeichnet. Hier werden sämtliche Einstellungen durchgeführt, welche die zu installierenden Clients betreffen, sowie benutzerdefinierte Scripts die nach der eigentlichen Installation aufgerufen werden definiert. Alle Grundlegenden Dateien, inklusive Beispielkonfigurationen werden bei der Installation des FAI-Servers angelegt.

Weitere Beispielkonfigurationen sind im Ordner „/usr/share/doc/fai-doc/examples/simple/“ zu finden. Das weitere Einrichten des Configspace wurde an Hand eines Tutorials (http://wiki.fai-project.org/wiki/UbuntuJauntyInstallationHowTo#Building_your_configspace) aus der offiziellen Wiki des FAI-Projekts durchgeführt.

```
projekt@VADER01:/srv/fai$ ls  
config  nfsroot  
projekt@VADER01:/srv/fai$ cd config/  
projekt@VADER01:/srv/fai/config$ ls  
basefiles  debconf      files  package_config  tests  
class      disk_config  hooks  scripts
```

Abbildung 7: Configspace

Inhalte der jeweiligen Unterordner des Configspace:

- **basefiles:** Tar-Archive mit der grundlegenden Verzeichnisstruktur der jeweils zu installierenden Distribution (bsp. „/lib“, „/bin“).
- **class:** Hier werden die Classes definiert, an Hand derer die Konfiguration der jeweils zu installierenden Client-Systeme erfolgt. Genauere Beschreibung, siehe folgende Seite.
- **debconf:** Grundlegende Konfigurationen, z.B. Einstellungen der Shell (Sprache etc.), Tastaturlayout.
- **disk_config:** Konfiguration der Partitionen und Dateisysteme, sowie bei Bedarf Software-Raids und LVM-Volumes.
- **files:** Beliebige Dateien für die Scripts aus dem „scripts“ Ordner, die mittels fcopy-Befehlen der jeweiligen Scripts auf das Client-System kopiert werden.
- **hooks:** Scripts zur Anpassung bzw. Erweiterung des FAI-Installationsvorgangs.
- **package_config:** Enthält Listen mit den zu verwendenden Packet-Management Tools und zu installierenden Paketen (auch Kernel).
- **scripts:** Benutzerdefinierte oder obligatorische Scripts (z.B. Grub-Installation) die nach Abschluss des eigentlichen Installationsvorgangs (Partitionierung, Installation der Pakete, etc.) ausgeführt werden.
- **tests:** Scripts für automatische Tests die nach Abschluss der Installation durchgeführt werden (z.B. Prüfung ob Installation erfolgreich).

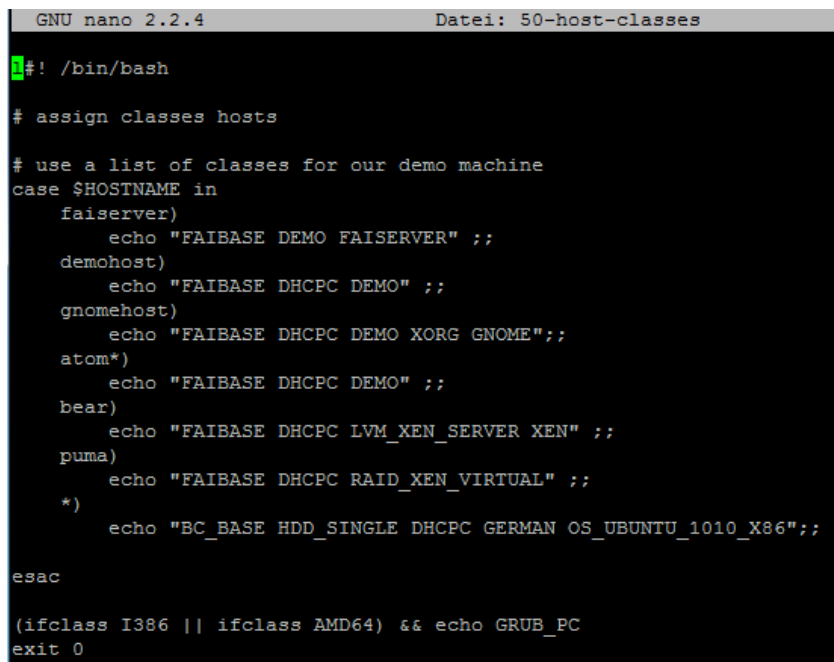
Auf den folgenden Seiten sollen alle Unterordner genauer beschrieben werden, in denen zusätzliche benutzerdefinierte Änderungen durchgeführt wurden. Alle nicht genauer beschriebenen Unterordner des Configspace wurden entweder ausschließlich nach dem auf der vorherigen Seite erwähnten Tutorial verändert oder in der jeweiligen Ursprungsform belassen.

basefile

Um das Basefile zu erstellen wurde zuerst auf einem der weiteren Server Ubuntu 10.10 Server installiert. Das eigentliche .tar Archiv wurde dann an Hand der Anleitung unter http://wiki.fai-project.org/wiki/UbuntuJauntyInstallationHowTo#Create_a_custom_base.tgz erstellt (Parameter entsprechend Ubuntu 10.10 angepasst) und in einem Unterordner „OS_UBUNTU_1010_X86“ (Bedeutung, siehe nächster Abschnitt) abgelegt.

class

Die zentrale Konfigurationsdatei des Configspace ist die Datei „50-host-classes“ im Verzeichnis „class“. Dort werden die sogenannten Classes definiert. Classes bezeichnen im Grunde Ordner, die innerhalb der jeweiligen Unterordner des Configspace angelegt werden können und je nach Definition in „50-host-classes“ während der Installation der Clients aufgerufen werden.



```
GNU nano 2.2.4      Datei: 50-host-classes
#!/bin/bash

# assign classes hosts

# use a list of classes for our demo machine
case $HOSTNAME in
    faiserver)
        echo "FAIBASE DEMO FAISERVER" ;;
    demohost)
        echo "FAIBASE DHCP DEMO" ;;
    gnomehost)
        echo "FAIBASE DHCP DEMO XORG GNOME";;
    atom*)
        echo "FAIBASE DHCP DEMO" ;;
    bear)
        echo "FAIBASE DHCP LVM_XEN_SERVER XEN" ;;
    puma)
        echo "FAIBASE DHCP RAID_XEN_VIRTUAL" ;;
    *)
        echo "BC_BASE HDD_SINGLE DHCP GERMAN OS_UBUNTU_1010_X86";;
esac

(ifclass I386 || ifclass AMD64) && echo GRUB_PC
exit 0
```

Abbildung 8: /srv/fai/config/class/50-host-classes

Wie man sieht, werden die Classes abhängig von den jeweiligen Hostnamen der zu installierenden Clients definiert. Die Hostnamen müssen dabei allerdings in der Konfiguration des DHCP-Servers den entsprechenden MAC-Adressen der jeweiligen Clients zugeordnet werden. Soll eine Vielzahl von Clients auf identische Art und Weise installiert werden (Ziel dieses Projekts), ist daher eine Definition der für die Client-Installation zu verwendenden Classes im Default-Abschnitt („*“ am Ende der Liste), einer Host-spezifischen Definition vorzuziehen. Die Auswahl der installierbaren Clients wird bei der hier angewandten Lösung ohnehin durch ein VLAN eingeschränkt (siehe [Punkt 2.2.1](#)). Die Class „OS_UBUNTU_1010_X86“ soll dabei die spezifischen Konfigurationen für das im Rahmen dieses Projekts zu installierende Ubuntu-Server 10.10 enthalten. Alle weiteren Definitionen, außer die des Default-Abschnitts waren bereits vorhandene Demo-Definitionen.

Hierbei ist schon der erste grobe Fehler unterlaufen. Wie man sieht ist bei allen Demo-Definitionen die Class FAIBASE definiert. In der benutzerdefinierten Default Definition die bei der Installation aller Test-Clients verwendet wird, jedoch nicht. In der gleichnamigen Datei „FAIBASE.var“ werden jedoch essenziell wichtige Parameter wie z.B. das root-Passwort des zu installierenden Clients oder das zu verwendende Partitionierungs-Tool definiert.

```
# root password for the new installed linux system; md5 and crypt are possible
# pw is "fai"
ROOTPW='$!$kBnWcO.E$djxB128U7dMkr1tJHPf6d1'

# MODULESLIST contains modules that will be loaded by the new system,
# not during installation these modules will be written to /etc/modules
# If you need a module during installation, add it to $kernelmodules
# in 20-hwdetect.source. But discover should do most of this job
MODULESLIST="usbkbd ehci-hcd ohci-hcd uhci-hcd usbhid psmouse"

# erros in tasks greater than this value will cause the installation to stop
STOP_ON_ERROR=700

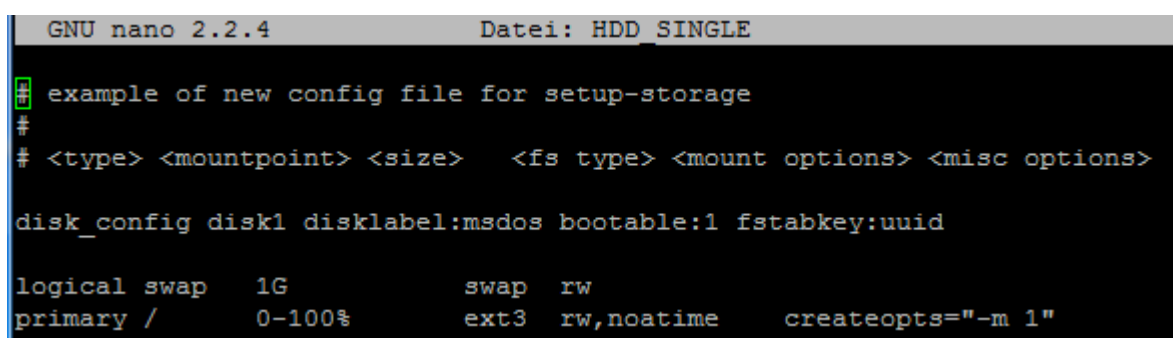
# use the new partitioning tool
USE_SETUP_STORAGE=1
```

Abbildung 9: Auszug aus FAIBASE.var

Beim ersten Testlauf der Client-Installation wurde daher, „FAIBASE.var“ nicht aufgerufen, damit der Parameter „USE_SETUP_STORAGE=1“ (Verwende setup-storage zur Partitionierung) nicht definiert und ohne diesen Parameter das ursprüngliche Partitionierungstool „setup_harddisks“ aufgerufen. Dieses benötigt jedoch eine andere Syntax als die im Folgenden Abschnitt verwendete. Somit schlugen die Partitionierung und damit auch die Client-Installation fehl. Der Fehler wurde erst nach langwieriger Suche behoben. Dabei wurden jedoch anstatt die Class „FAIBASE“ wie bei allen anderen Class-Definitionen in „50-host-classes“ auch in der Default-Definition hinzuzufügen, an Hand des Tutorials die Parameter „USE_SETUP_STORAGE“ und „ROOTPW“ einfach in die Datei „BC_BASE.var“ im Ordner „class“ eingetragen. Der eigentliche Fehler wie er hier beschrieben wurde, also das Fehlen der Definition der Class „FAIBASE“ wurde erst im Rahmen dieser Dokumentation erkannt.

disk_config

Die Partitionierung während der Client-Installation erfolgt mit dem Fai-eigenen Tool „Setup-storage“ (<http://wiki.fai-project.org/wiki/Setup-storage>). Es basiert auf dem Partitionierungstool „parted“ und besitzt eine eigene Syntax zur Definition der Partitionen und Dateisysteme, welche im Ordner „disk_config“ je nach Class definiert wird.



```
GNU nano 2.2.4 Datei: HDD_SINGLE
# example of new config file for setup-storage
#
# <type> <mountpoint> <size> <fs type> <mount options> <misc options>
disk_config disk1 disklabel:msdos bootable:1 fstabkey:uuid
logical swap 1G swap rw
primary / 0-100% ext3 rw,noatime createopts="-m 1"
```

Abbildung 10: /srv/fai/config/disk_config/HDD_SINGLE

Als Swap-Partition wird eine logische Partition mit einer Größe von 1GB definiert. Den restlichen Speicherplatz (0-100%) nimmt die primäre root Partition, mit ext3 Dateisystem ein.

files

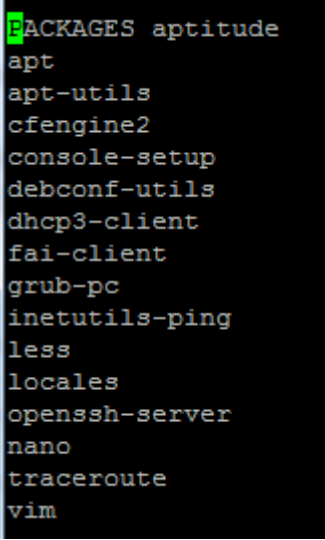
Hier wurden Zusätzlich zu den im Tutorial definierten Files nachträglich noch die im Rahmen dieses Projekts erstellten Python Scripts Pywebback und Pywiki_install abgelegt. Allerdings wurde die Integration dieser in die FAI-Lösung vernachlässigt und nicht getestet. Dies kann man aber unter der Devise „nice to have“ abhaken, da in der Zielsetzung des Projekts ([Punkt 1.2](#)) explizit definiert wurde, das die Scripts unabhängig von der FAI-Lösung lauffähig sein sollen.

package_config

Hier wurden zuerst die Standardpaketlisten aus dem Tutorial definiert. Zusätzlich sollten noch einige weitere rudimentäre Pakete installiert werden, z.B. inetutils-ping (ping befehl), traceroute, nano. Dabei gilt es zu beachten das zu Beginn anstatt dem Paket „grub-pc“ (Grub2, Standard Bootloader seit Ubuntu 10.04), das Paket „grub“ (Grub1) eingetragen wurde. Daher war der nächste Versuch einer Client-Installation, mit der Ausnahme diverser Grub Fehlermeldungen zwar erfolgreich, jedoch konnte der installierte Client nicht gebootet werden, da kein Bootloader vorhanden war.

Die eigentliche Grub Installation wird nach allen Paketinstallationen durch das Script „10-setup“ im Ordner „scripts“ durchgeführt (Siehe folgender Abschnitt scripts). Hier wurde das richtige Beispielscript für die Installation von Grub2 ausgewählt, welches jedoch inkompatibel zu Grub1 ist. Zu Beginn war jedoch völlig unklar das es sich bei der installierten Grub Version um Grub1 handelt und es wurde davon ausgegangen das der Fehler im Installations-Script „10-setup“ liegt. Daher wurde das Script in aufwändiger „Trial and Error“ Manier (nach jeder Anpassung neuer Installationsversuch) angepasst, bis die Installation von Grub1 erfolgreich ablief. Beim Booten des Clients mit dem funktionierenden Grub Bootloader stach es jedoch sofort ins Auge, das es sich dabei um Grub1 handelt, welcher natürlich standardmäßig nicht mit Ubuntu 10.10 verwendet werden sollte.

Nun war es daran das passende Paket für Grub2 zu finden. Nach einem „apt-cache search grub“ und einer Analyse der gefundenen Grub-Pakete mit „apt-cache show“, war schnell klar, dass es sich dabei um das Paket „grub-pc“ handelt.



```
PACKAGES aptitude
apt
apt-utils
cfengine2
console-setup
debconf-utils
dhcp3-client
fai-client
grub-pc
inetutils-ping
less
locales
openssh-server
nano
traceroute
vim
```

Abbildung 11: Auszug aus
package_config/BC_BASE

scripts

Alle Scripts werden erst nach Partitionierung, Kopieren der Basefiles und Installation der Pakete ausgeführt.

Auch hier wurde die Einrichtung hauptsächlich wieder an Hand des Tutorials durchgeführt. Allerdings musste die Datei „10-setup“ im Unterordner „GRUB_PC“ noch angepasst werden:

```
#rootdelay für dell server in /etc/default/grub setzen
$ROOTCMD sed -i "s/GRUB_CMDLINE_LINUX=.*\/GRUB_CMDLINE_LINUX=\"rootdelay=40\"/" /etc/default/grub

#konsolen auflösung in /etc/default/grub setzen
$ROOTCMD sed -i "s/#GRUB_GFXMODE=.*\/GRUB_GFXMODE=1024x768x16/" /etc/default/grub

$ROOTCMD sed -i "s/set gfxmode=\${GRUB_GFXMODE}\/set gfxmode=\${GRUB_GFXMODE} \nset gfxpayload=keep/" /etc/grub.d/00_header
$ROOTCMD update-grub
```

Abbildung 12: Auszug aus scripts/GRUB_PC/10-setup

Das Voranstellen der Shell-Variablen „\$ROOTCMD“ führt dazu, dass der nachfolgende Befehl in der beim Aufruf der Scripts schon weitestgehend fertig installierten Client-Umgebung ausgeführt wird. Unerlässlich war das Setzen des Kernel-Parameters „rootdelay=40“. Die verwendeten Dell-Server haben ein bekanntes Problem mit dem SCSI-Controller der Festplatten (Siehe: <http://web.archiveorange.com/archive/v/PqQ0RFzqdR4yP8ciRKag>). Wird der Parameter „rootdelay“ nicht gesetzt, bricht der Bootvorgang mit der Fehlermeldung „Gave up waiting for root device“ ab. Die beiden weiteren benutzerdefinierten Parameter „GRUB_GFXMODE=...“ und „set_gfxmode“ aktivieren den Framebuffer und setzen die Konsolenauflösung auf 1024x768 mit 16Bit Farbtiefe, was das Arbeiten an der Konsole angenehmer macht und von praktisch jedem noch im Einsatz befindlichen Farbmonitor unterstützt werden dürfte.

Zusätzlich wurden noch zwei weitere Scripts definiert, „60-useradd“ und „70-hostname“:

```
projekt@VADER01:/srv/fai/config/scripts/BC_BASE$ ls
10-misc 20-fcopy 30-interface 50-removable_media 60-useradd 70-hostname
```

Abbildung 13: Benutzerdefinierte Scripts

Das Script „60-useradd“ enthält einen einfachen „useradd“ Befehl, zum Anlegen eines standardmäßigen Users ohne besondere Rechte. Das andere Script „70-hostname“ generiert über den Befehl „\$ROOTCMD sed -i \"s/.*\/FAI\$RANDOM/\" /etc/hostname“ einen zufälligen Hostnamen für den zu installierenden Client (z.B. FAI123).

2.3.5 Finale Konfiguration und Tests

Der Befehl „fai-chboot“ erstellt die PXE-Boot Konfiguration für die zu installierenden Clients. Im Projekt wurde der Befehl mit den Parametern „fai-chboot -IB default“ (mögliche Parameter siehe: <http://fai-project.org/doc/man/fai-chboot.html>) nach Beispiel des im vorherigen Abschnitt bereits mehrfach erwähnten Tutorials aufgerufen. Der Parameter „default“ steht dabei für alle Clients die vom FAI-Server booten wollen.

```
root@VADER01:~# fai-chboot -IBv default
Booting kernel vmlinuz-2.6.32-5-486
  append initrd=initrd.img-2.6.32-5-486 ip=dhcp
    FAI_FLAGS=verbose,sshd,reboot

default has no IP in hex default
Writing file /srv/tftp/fai//pxelinux.cfg/default for default
```

Abbildung 14: fai-chboot (-IBv für verbose)

Die „FAI_FLAGS“ (standardmäßig durch Parameter –B gesetzt) definieren das Verhalten des Clients während der Installation. Dabei steht „verbose“ für die ausführliche Ausgabe der Abläufe des Installationsvorgangs (zusätzlich „debug“ wäre noch ausführlicher), „sshd“ startet während der Installation einen SSH-Daemon, welcher die Überwachung des Clients via SSH ermöglicht und „reboot“ sorgt dafür das der Client nach erfolgreicher Installation automatisch neu startet.

Wie im vorherigen Abschnitt erwähnt wurde, traten bei den ersten Client Installationen mehrere Fehler auf. Um diese zu erkennen muss man die entsprechenden Logdateien auf dem installierten Client unter „/tmp/fai“ *auswerten*. Dabei war zu Beginn kein Zugriff auf die Logdateien möglich, da der Client zwar nicht automatisch neustartet, dies jedoch beim Drücken von „strg + C“ um auf Konsole zu gelangen tut. Um Zugriff auf die Logdateien zu erlangen wurden stundenlang, die verschiedensten Versuche unternommen und dabei zahlreiche Fehleinschätzungen vorgenommen:

- Kein Zugriff über SSH probiert
- Unter „/etc/fai/fai.conf“ kann ein User definiert werden um die Logdateien der Clients auf dem zentralen Server abzulegen (nicht probiert)
- Mehrfach neue Client Installationen durchgeführt „um auf Nummer Sicher zu gehen“ (auch mit verschiedenen fai-chboot Parametern)

Dabei war der Start eines zweiten Terminals (z.B. „strg+alt+F2“), zum Zugriff auf die Logdateien ebenfalls nicht möglich, da eine weiteres Terminal für die Client-Installation in den Parametern von „fai-chboot“ definiert werden muss. Die Lösung sah dann am Ende so aus „fai-chboot -I -f verbose,sshd,debug,createvt default“. Der Parameter „createvt“ legt das zusätzliche Terminal an. Ein einfacher Blick in die Manpages hätte hier genügt um stundenlange Arbeit zu sparen.

Danach war es möglich über das zweite Terminal die Logdateien zu öffnen, die im vorigen Abschnitt erwähnten Probleme ausfindig zu machen und zu beheben. Mit Testinstallationen via PXE-Boot auf einem der Server und einer virtuellen Maschine (Virtual-Box unter Ubuntu) konnte dann die volle Funktionsfähigkeit der hier eingerichteten FAI-Lösung bestätigt werden. Das auf den Clients installierte Ubuntu-Server 10.10 war voll Einsatzfähig.

2.4 Fazit

2.4.1 Security Aspekte

Alle für die FAI-Lösung verwendeten Passwörter werden in den Konfigurationsdateien ausschließlich als Hashes (md5 oder Linux Befehl „crypt“) gespeichert. Somit ist es für Angreifer, auch bei uneingeschränktem Zugang auf den zentralen Server nicht ohne weiteres möglich die Passwörter auszulesen. Eine besondere Eigenschaft der hier implementierten Lösung ist die vollständige Abschottung des internen, für Client-Installationen verwendeten Netzes mittels VLAN und „iptables“ Firewall. Dies ermöglicht eine abgesicherte Risikoanalyse der installierten Clients, bevor man diese in die Produktiv-Umgebung aufnimmt. Zusätzlich wird ausgeschlossen das unbefugte Clients vom zentralen Server via PXE booten.

2.4.2 Mögliche Einsatzbereiche

Die möglichen Einsatzbereiche für eine FAI-Lösung werden schon in der Einleitung dieser Dokumentation ausführlich beschrieben, überall wo eine Vielzahl von Linux-Systemen auf identische Art und Weise installiert werden muss. Im Serverbereich können dies sowohl rudimentäre Installationen die danach für den individuellen Einsatzbereich des jeweiligen Servers manuell angepasst werden, als auch komplett fertige Serverinstallationen mit allen benötigten Paketen und Konfigurationen, z.B. bei einem Webhoster für Root-Server.

Im PC-Bereich ist eine FAI-Lösung allerdings am besten aufgehoben, da bei einem sinnvollen Domänenkonzept egal ob bei Schulen, Hochschulen, Firmen usw. Daten und Berechtigungen auf zentralen Fileservern und Domänencontrollern gespeichert werden (z.B. LDAP, Active-Directory). Alle Client-PCs werden identisch installiert, der jeweilige User kann dann unabhängig von welchem Client aus er sich anmeldet auf dieselben Daten und Dienste zugreifen. Mit Hilfe der Classes sind dann auch „Extrawürste“ d.h. vom Standard abweichende Client-Installationen möglich.

2.4.3 Allgemeines Fazit

Die implementierte FAI-Lösung ist voll funktionsfähig und bestens für die vorgesehenen Einsatzbereiche geeignet. Allerdings wurden im Verlauf dieses Teils des Projekts zahlreiche Fehler begangen, die zu einer förmlichen Explosion des zeitlichen Aufwandes geführt haben. FAI ist eine relativ komplexe und umfangreiche Software mit zahlreichen Funktionen, von denen hier nur ein Bruchteil auch wirklich genutzt wurde. Für eine effektive und weniger zeitaufwändige Implementierung von FAI ist daher zunächst eine intensive Einarbeitung in die Dokumentation, sowie die grundlegenden Funktionen und Konzepte erforderlich. Dies wurde hier komplett vernachlässigt. Es wurde einfach unter Zuhilfenahme unterschiedlicher Tutorials „drauflos“ konfiguriert und installiert. Selbst bei der Fehlersuche wurde oft an den falschen Stellen gesucht (stupides „googeln“ anstatt offizielle Doku und Manpages). Manche Fehler wurden gar erst beim Erstellen dieser Dokumentation entdeckt. Glücklicherweise wurde aus diesen Fehlern bei der im Folgenden beschriebenen Entwicklung der Scripts gelernt und zunächst Richtlinien definiert, welche es während der Programmierung zu beachten galt.

Merke für die Zukunft: **Planung ist bei einem Projekt das A und O!**

3. Richtlinien zur Script-Entwicklung

3.1. Programmiersprache Python

Alle Scripts sollen in Python implementiert werden. Die Sprache ist Interpreter basiert und zeichnet sich aus durch Plattformunabhängigkeit, eine leicht erlernbare jedoch enorm mächtige Syntax, sowie die Python-Shell. Die Python-Shell startet nach Aufruf des Interpreters ohne Parameter und interpretiert sämtlichen Quelltext der eingegeben wird in Echtzeit.

Dies eignet sich besonders gut während einer Lern- und Entwicklungsprozesses um z.B. das Verhalten von noch unbekannt Funktionen zu testen.

Daneben bietet Python noch zahlreiche integrierte Module, die besonders gut für Administrationsaufgaben und die Verwendung im Script-Bereich geeignet sind.

Ein gutes Beispiel ist hier das OS Modul welches Betriebssystemspezifische Funktionen, wie z.B. jegliche Art von Verzeichnishandling in Python abstrahiert.

Python Programme sind besonders intuitiv lesbar und erzwingen das saubere Einrücken des Quelltextes, da anders als bei vielen anderen Programmiersprachen der jeweilige Scope anstatt mit geschweiften Klammern explizit durch das Einrücken definiert wird.

Bei der Entwicklung soll ausschließlich Version 2.7 von Python zum Einsatz kommen.

Aktuell ist Version 3.2, jedoch gab es mit Version 3.0 einige Änderungen an der Syntax und so werden alle gängigen Linux-Distributionen aus Kompatibilitätsgründen noch standardmäßig mit der Version 2.7 ausgeliefert, welche auch weiterhin von offizieller Seite gepflegt wird. Offiziell gibt es den Python-Interpreter für Linux (wie bereits erwähnt in jeder gängigen Distribution standardmäßig vorhanden), Windows und Mac-OS X.

Python wird von der Python Software Foundation, einer non-profit Organisation unter einer eigenen Open-Source Lizenz ohne Copyleft-Prinzip herausgegeben.

Nahezu alle Python-Versionen, auch die in diesem Projekt verwendete, sind jedoch kompatibel zur GPL und lassen sich somit bedenkenlos zur Programmierung von GPL-lizenzierter Software verwenden.

3.2 Aufbau der Scripts

Module

Die Scripts sollen mehrere Module beinhalten, mit verschiedenen für die Funktionalität des Hauptscripts benötigten Funktionen.

Alle Funktionen sollen vollständig unabhängig voneinander implementiert werden, um eine maximale Wiederverwendbarkeit des Codes zu gewährleisten. Zur besseren Strukturierung des Quellcodes sollen jedoch verwandte Funktionen innerhalb eines Moduls zusammengefasst werden.

Um unnötige Programmierarbeit zu vermeiden und eine bestmögliche Kompatibilität aller Scripts zu gewährleisten, soll bei der Implementierung der eigenen Module möglichst vollständig auf bereits in Python integrierte Module zurückgegriffen werden.

Hauptscript

Das jeweilige Hauptscript soll mittels einer ebenfalls in einem separaten Modul implementierten Funktion eine Konfigurationsdatei importieren.

An Hand der Parameter die in der Konfigurationsdatei definiert wurden, sollen dann die für die eigentliche Funktionalität des Hauptscripts benötigten Funktionen aufgerufen werden.

3.3 Grundlegende Richtlinien

Für die Entwicklung der Scripts solle Eclipse mit dem Plugin „Pydev“ verwendet werden. Eclipse ist bereits aus den bisherigen Softwareentwicklungs-Vorlesungen bekannt und erfordert daher keine Einarbeitungszeit.

Kommentare sollen nur dort eingefügt werden, wo immer der Quellcode nicht intuitiv lesbar ist oder es die Komplexität des Codes erforderlich macht.

Das jeweilige Hauptsript soll Exception-Handling beinhalten und eine Logdatei erstellen, in der alle während dem Ablauf des Scripts aufgetretenen Fehler möglichst ausführlich und gut verständlich angezeigt werden.

3.4 Ablauf der Scriptentwicklung

- 1) **Grundlegende Funktionsweise definieren**
Was soll das Script überhaupt machen?
- 2) **Bedarf ermitteln**
Welche Funktionalität wird benötigt?
- 3) **Funktionen an Hand des ermittelten Bedarfes implementieren und Testen**
- 4) **Hauptsript implementieren und testen**
Funktionen falls nötig anpassen oder neue Funktionen implementieren.
- 5) **Exception-Handling und Logging in Hauptsript implementieren**
- 6) **Finale Tests**
z.B. Fehler provozieren um Effektivität und Korrektheit des zuvor implementierten Logging und Exception-Handling zu testen.

4. Pywebback

4.1 Grundlagen

Die eigentliche Funktionsweise ergibt sich aus dem Quelltext, sowie der beiliegenden Readme Datei. Diese geht unter anderem auf die Systemvoraussetzungen des Scripts ein und enthält alle wichtigen Hinweise zu dessen ordnungsgemäßer Verwendung.

Daher soll in diesem Abschnitt nur auf die grundlegende Funktionalität von Pywebback und dessen Module eingegangen, sowie ein abschließendes Fazit gezogen werden.

4.1.1 Grundlegender Aufbau

Pywebback soll speziell auf die Sicherung von Webanwendungen ausgelegt sein. Das heißt, es werden folgende Bestandteile gesichert:

- Dateien der Webanwendung im Webroot (spezifische Auswahl einzelner Verzeichnisse möglich)
- Datenbank der Webanwendung
- Konfiguration des Webservers

Moderne Webanwendungen (primär Content-Management-Systeme) setzen sich maßgeblich aus den jeweiligen Dateien (php, html usw.) und einer dazu gehörenden Datenbank zusammen. Daraus resultiert die Trennung von Inhalt und Gestaltung der Webanwendung.

Zusätzlich soll noch die Konfiguration des Webservers gesichert werden, da hier auch viele bedeutsame anwendungsspezifische Einstellungen gemacht werden können, wie z.B. url-rewriting oder die Blockierung bestimmter Verzeichnisse.

Bei einem regelmäßigen Aufruf des Scripts (z.B. wöchentlich) ist Pywebback dazu in der Lage mehrere Backup-Dateien, je nach definierter Anzahl zu verwalten.

Ist die definierte Anzahl an Backup-Dateien erreicht, wird die jeweils älteste Datei überschrieben. Diese Funktionalität wird mittels der Funktion „rotate“ aus dem Modul „rotate.py“ (siehe [Punkt 4.2.4](#)) realisiert.

Bei einem wirkungsvollen Backup ist es natürlich wenig sinnvoll dieses auf demselben System zu speichern, auf dem die zu sichernde Webanwendung läuft.

Um das Backup extern zu speichern bieten sich daher unter anderem folgende Optionen an:

- Sicherung auf einem Fileserver (z.B. via NFS oder SMB)
- Sicherung auf einem FTP-Server
- Sicherung auf einer externen Festplatte

Für Pywebback wurde eine FTP-basierte Lösung ausgewählt, da viele Webhosting-Provider einen gesonderten Backup-Speicherplatz anbieten auf den via FTP zugegriffen werden kann oder bei einer Migration zu einem neuen Webhosting Anbieter, eine direkte Sicherung auf den neuen Webspace möglich wird. Die Sicherung auf einen Fileserver eignet sich in diesem Fall vor allem bei einem Einsatz im Rechenzentrum, in dem ein solcher bereits für ähnliche Zwecke vorhanden ist. Sollte dennoch eine Sicherung via NFS- oder SMB-Protokoll erwünscht sein, ließe sich diese auf Grund des vollständig modularen Aufbaus von Pywebback mit nur minimalen Änderungen am bereits bestehenden Script implementieren.

Die Sicherung auf einer externen Festplatte lässt sich auch ohne zusätzliche Funktionen realisieren, da diese bei allen halbwegs modernen Betriebssystemen automatisch nach dem Anschließen ins Filesystem eingebunden wird. Anschließend muss man nur noch das gewünschte Verzeichnis auf der externen Festplatte als Backup-Verzeichnis in der Konfigurationsdatei angeben.

4.2 Implementierung der Module

Es ist ratsam beim Lesen des folgenden Abschnittes gleichzeitig den Quellcode der jeweiligen Module zu öffnen, um die nachfolgenden Ausführungen besser verstehen zu können. Alle für die Verwendung von Pywebback benötigten Module befinden sich im Package „pywebback_modules“.

4.2.1 Modul „backupdb“

Die Funktion „backupdb“ im Modul „backupdb“ verwendet das Tool „mysqldump“, des MySQL-Servers zur Sicherung der als Parameter übergebenen Datenbank.

Der Aufruf von im Betriebssystem verfügbaren Befehlen erfolgt über die Funktion „Popen“ des Moduls „Subprocess“. Durch den Aufruf von „Popen()“ wird ein neuer Prozess erstellt, daher muss um einen reibungslosen Ablauf des Scripts zu gewährleisten der Aufruf von „Popen()“ über die Funktion „Popen.wait()“ erfolgen.

„Popen.wait()“ hält den Ablauf des restlichen Scripts an, bis der an „Popen.wait()“ übergebene Prozess beendet wurde. Dies ist notwendig, da ansonsten sofort nach Aufruf von „mysqldump“ ohne, dass die Sicherung bereits abgeschlossen wäre, im Hauptsript die Funktion „ftpdconnect“ zum Abbau der FTP-Verbindung aufgerufen würde.

Sollte „Popen()“ nach der Ausführung des aufgerufenen Befehls eine 1 zurückliefern (Code für Fehler) wirft „backupdb“ eine Exception, da die Datenbanksicherung in diesem Fall fehlgeschlagen ist.

Die Funktion „restoredb“ dient analog zu „backupdb“ zur Wiederherstellung der Datenbank. Genauer gesagt wird „restoredb“ ein Verzeichnis als String übergeben, aus dem es alle Dateien heraus sucht die auf „.sql“ enden und die erste gefundene Datei wiederherstellt.

4.2.2 Modul „configimport“

Die Funktion „confimport“ im Modul „configimport“, dient zum Importieren der Konfigurationsdatei, unter Verwendung des in Python integrierten Moduls „ConfigParser“.

„Confimport“ liest alle Attribute und deren Werte aus der übergebenen section (in der jeweiligen Konfigurationsdatei definiert durch eckige Klammern), der übergebenen Konfigurationsdatei ein und liefert diese als Dictionary zurück.

Ein Dictionary ist ein Python spezifische Datentyp von Key-Value Paaren. Analog zu einem Array spricht man das Dictionary über das jeweilige Key-Attribut an, um den entsprechenden Wert (Value) zu erhalten. Als Beispiel liefert „dictionary[„db-name“]“ den Wert, der in der Konfigurationsdatei unter dem Attribut „db-name“ eingetragen wurde.

Ein Dictionary kann Werte mit beliebigen Datentypen, auch weitere Dictionaries oder gemischte Datentypen enthalten, die jeweiligen Keys müssen jedoch immer vom Typ String sein. Im Fall von „confimport“, sind auch alle aus der Konfigurationsdatei in das Dictionary importierten Werte Strings.

4.2.3 Modul „ftp“

Im Modul „ftp“ wird zuerst ein zufälliges Verzeichnis im Home-Verzeichnis des Benutzers, von dem das Script aufgerufen wurde erstellt.

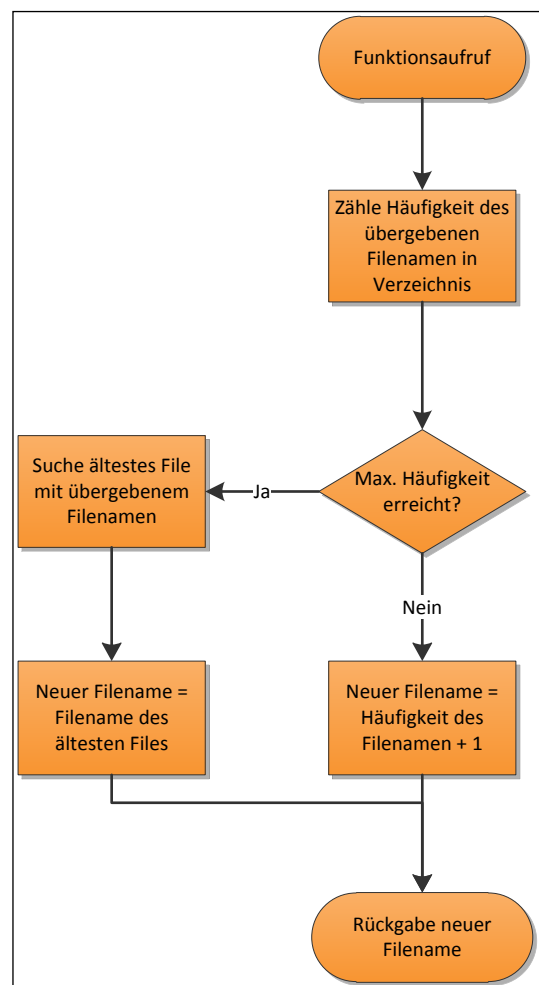
In dieses Verzeichnis mountet die Funktion „ftpconnect“ mit dem Linux-Programm „curlftpfs“ einen beliebigen FTP-Server. Curlftpfs bietet im Vergleich zu einem Standard-Mount-Befehl den Vorteil, dass es sich ohne Administratorrechte benutzen lässt. Nach Einbinden des FTP-Servers liefert „ftpconnect“ den Namen des zuvor erstellten zufälligen Verzeichnisses zurück, damit das Hauptscript mittels der Backup-Funktionen dazu in der Lage ist die Backup-Dateien dort zu speichern.

Die Funktion „ftpdconnect“ trennt die Verbindung, in dem sie das FTP-Verzeichnis unmounted und löscht danach das zufällige Verzeichnis.

4.2.4 Modul „rotate“

Die Funktion „rotate“ lässt sich am besten an Hand eines Beispiels erklären:

- 1) `rotate („www_bak“, „/bakdir“, 3)`
Aufruf der Funktion rotate mit Dateiname „www_bak“, Verzeichnis in dem die Backups liegen „/bakdir“ und maximale Anzahl der Backups 3.
- 2) Überprüfe wie oft „www_bak“ in „/bakdir“ vorkommt, unabhängig von Dateiendung oder Nr. am Ende.
- 3) Wenn Bereits 3 Dateien die den Namen „www_bak“ enthalten in „/bakdir“ vorhanden sind („www_bak1“ usw.) suche älteste dieser 3 Dateien und liefere Namen der ältesten Datei als Ergebnis zurück.
Pywebback überschreibt die älteste Datei
- 4) Wenn z.B. erst 2 Dateien die den Namen „www_bak“ enthalten in „/bakdir“ vorhanden sind, liefere Namen „www_bak“ + Anzahl bereits vorhandenen Dateien (also „www_bak3“) zurück.
Pywebback erstellt neue Datei „www_bak3“.



Die Funktion „rotate“ läuft wie im Kapitel „Richtlinien der Script-Entwicklung“ definiert vollkommen unabhängig von Pywebback ab.

Der Hauptvorteil der sich daraus ergibt ist, dass die Namen neuer Backup-Dateien unabhängig von der jeweiligen Funktion welche die Backups erstellt variiert werden können und diese Funktionalität somit nur an einer Stelle implementiert werden musste. Die jeweiligen Funktionen zum Erstellen der Backups erhalten beim Aufruf nur den Dateinamen, den sie für das Erstellen der Backups verwenden sollen und bleiben somit auf ihre grundlegende Funktionalität beschränkt.

4.2.5 Modul „tarbackup“

Die Funktion „tarbak“ aus dem Modul „tarbackup“ ist zuständig für die Sicherung von Dateien. Sie verwendet die in Python integrierte Funktion „tarfile“ um „.tar“ Archive der jeweils übergebenen Verzeichnisse zu erstellen.

Dabei wird über die Variable „tobackup“ eine mittels Semikolon getrennte Menge von Unterordnern oder Dateien innerhalb des zu sichernden Verzeichnisses „dir“ übergeben, die mit Hilfe der Funktion „split“ des String Datentyps getrennt und danach dem „.tar“ Archiv hinzugefügt wird.

Ist „tobackup“ leer, was als Standard definiert ist falls beim Aufruf von „tarbak“ kein Wert für „tobackup“ übergeben wird, sichert „tarbak“ einfach das gesamte in der Variable „dir“ übergebene Verzeichnis, inklusive aller Unterordner.

Die Funktion „restoretar“ entpackt ein beliebiges „.tar“ Archiv, zur Wiederherstellung der Sicherung.

4.3 Implementierung des Hauptscripts

Pywebback legt zu Beginn mit Hilfe des Python Moduls „logging“ eine Logdatei in der später alle Exceptions, sowie alle erfolgreichen Vorgänge des Scripts eingetragen werden.

Danach wird mit der Funktion „confimport“ die Konfigurationsdatei eingelesen und an Hand der dort definierten Parameter die jeweils benötigten Funktionen zum Erstellen der Sicherungen aufgerufen.

Erwähnenswert ist hier das sich in Python Exception-Handling und Logging wunderbar kombinieren lassen. Auf die Funktionsweise von Exception-Handling an sich soll hier jedoch nicht genauer eingegangen werden, da dies auf dieselbe Art und Weise wie in anderen gängigen Programmiersprachen (z.B. Java) funktioniert.

Der Befehl um in Python eine Exception abzufangen nennt sich anstatt „catch“ jedoch „except“, der Befehl zum Werfen von Exceptions „raise“.

Wird die Funktion „logging.exception“ nach Auftreten einer Exception im „except“ Block aufgerufen, schreibt diese einen als String übergebene Nachricht, sowie den Originaltext der aufgetretenen Exception in die jeweilige Logdatei.

Beim Aufruf von „raise“ aus dem „except“ Block wird die aufgetretene Exception weitergeleitet. Im Fall von Pywebback wird die Ausführung des Scripts dadurch vollständig abgebrochen, da die Exception nirgendwo mehr abgefangen wird.

Dies geschieht in Pywebback unter folgenden Umständen:

- Zu Beginn definierte Konfigurationsdatei existiert nicht.
- Importieren der Konfigurationsdatei schlug fehl.
- Verbindung zu FTP-Server (nur falls in Konfigurationsdatei definiert) schlug fehl.
- Verzeichnis in dem die Backup-Dateien abgelegt werden sollen existiert nicht und kann auch nicht angelegt werden.

Schlägt eine der Backup-Funktionen fehl, wird dies lediglich in der Logdatei vermerkt. Das Script läuft jedoch trotzdem weiter und führt weitere, eventuell nachfolgende Backup-Funktionen aus.

4.4 Webrestore

Das dem Gesamtpaket von Pywebback beiliegende Script Webrestore dient zur Wiederherstellung der mit Pywebback gesicherten Dateien. Dabei verwendet Webrestore, die im vorigen Abschnitt beschriebenen Wiederherstellungsfunktionen der jeweiligen Module. Webrestore verwendet die FTP-Funktionen um Daten auf einen FTP-Server wiederherzustellen. Bei Anwendung von Webrestore ergibt sich auch ein großer Nachteil des Moduls „backupdb“ (siehe [Punkt 4.2.1](#)). Der in „backupdb“ zur Sicherung der Datenbank verwendete Befehl „mysqldump“ sichert nur alle Tabellen der jeweiligen Datenbank und deren Inhalte, die SQL-Befehle zum Anlegen der Datenbank fehlen in der resultierenden SQL-Datei. Daher ist es für die Wiederherstellung der Inhalte einer Datenbank zwingend notwendig die Datenbank auf dem Ziel-MySQL-Server erst anzulegen.

4.5 Fazit

4.5.1 Security-Aspekte

Pro

Für den Backup-Vorgang werden nur minimale Rechte benötigt. Das heißt lesender Zugriff auf die zu sichernden Verzeichnisse und die zu sichernde Datenbank, sowie Schreibrechte auf das Verzeichnis in dem die Backups abgelegt werden sollen und auf das Hauptverzeichnis des Scripts, zum Anlegen der Logdatei.

Der Pfad zum Anlegen der Logdatei kann jedoch auch zu Beginn des Hauptscripts definiert werden. Eine Alternative wäre es hier, den Pfad für die Logdatei in der Konfigurationsdatei festzulegen, allerdings stoßen wir dabei auf das „Henne – Ei“ Problem.

In der Logdatei soll auch mitgeloggt werden, ob die Konfigurationsdatei nicht eingelesen werden kann. Wird der Pfad in dem die Logdatei abgelegt werden soll jedoch erst in der Konfigurationsdatei definiert, existiert die Logdatei beim Laden der Konfigurationsdatei noch gar nicht. Durch die Verwendung von „curlftpfs“ zum Mounten eines FTP-Verzeichnisses werden auch dafür keine Administrator-Rechte benötigt.

Contra

Für die Nutzung der Tools „curlftpfs“ und „mysqldump“ zum Aufbau einer FTP-Verbindung bzw. Sichern der Datenbank müssen die entsprechenden Passwörter in Klartext, in der Konfigurationsdatei abgelegt werden. Beide Tools unterstützen beim Aufruf keine gehashten Passwörter. Eine Möglichkeit wäre es, die Passwörter mit einer Hash-Funktion in der Konfigurationsdatei zu speichern, aus der man die Passwörter innerhalb des Scripts wieder in das ursprüngliche Klartext-Format zurückrechnen kann. Allerdings würde dies nur den Anschein von zusätzlicher Sicherheit wahren, da man den Hash, sollte man ihn im Programm wieder in Klartext umwandeln können auch mit einem x-beliebigen Tool aus dem Internet wieder zurückrechnen kann. Daher wurde bei Pywebback darauf verzichtet.

Bei einem einmaligen Aufruf des Scripts kann man die Konfigurationsdatei bzw. nur die Passwörter natürlich sofort wieder löschen, die Passwörter wären sicher. Die grundlegende Absicht dieses Projektes ist es jedoch die Scripts vollständig automatisiert ablaufen zu lassen, im Fall von Pywebback z.B. für eine wöchentliche Sicherung. Aus diesem Grund ist es zwingend notwendig die Konfigurationsdatei, inklusive der Passwörter auf demselben System wie das Script zu halten. Das System auf dem Pywebback verwendet wird sollte daher gut abgesichert sein, was bei einem System auf dem ein Webserver mit Zugriffen von außen läuft ohnehin eine Grundvoraussetzung ist.

Da trotz allen Sicherheitsmaßnahmen immer die Möglichkeit besteht, dass ein System kompromittiert wird, sollten folgenden Maßnahmen ergriffen werden um den Schaden durch das Bekanntwerden der in der Konfigurationsdatei eingetragenen Passwörter so gering wie möglich zu halten:

- Anlegen eines separaten Benutzers für die Datenbank-Sicherung, der ausschließlich Leserechte auf die zu sichernde Datenbank besitzt.
- Sicherung via FTP nur bei manuellem Aufruf, nicht wenn das Script automatisiert gestartet wird, Passwort nach Abschluss der Sicherung sofort aus Konfigurationsdatei löschen.

4.5.2 Mögliche Einsatzbereiche

Pywebback soll im Allgemeinen nur eine Beispiel-Implementierung zur Automatisierung einer Administrations-Aufgabe für Webanwendungen unter Verwendung von Python darstellen (siehe [Punkt 1.2 Zielsetzung](#)). Für den automatisierten, produktiven Einsatz z.B. in einem Rechenzentrum gibt es mit Sicherheit eine Vielzahl besser geeigneter, flexiblerer und bewährter Backup-Lösungen.

Pywebback hat dennoch seine Daseinsberechtigung. Ein mögliches Anwendungsszenario wäre zum z.B.:

Man will eine oder mehreren Webanwendungen vom einen Web-Hosting-Anbieter (Voraussetzung: Shell-Zugriff auf Web-Hosting z.B. über SSH) zu einem anderen Anbieter migrieren. Mit Pywebback kann man ohne zusätzliche Kosten und mit wenig Konfigurationsaufwand schnell eine Sicherung aller wichtigen Bereiche seiner Webanwendung erstellen. Hat man sogar schon FTP-Zugang zu seinem neuen Webspace, kann man diese Archive ebenso schnell und unkompliziert dort wiederherstellen.

4.5.3 Allgemeines Fazit

Pywebback stellt eine simple und leicht anpassbare, weitestgehend plattformunabhängige Backup-Lösung für Webanwendungen dar. Die Richtlinien zur Script-Entwicklung aus Abschnitt 1.3 wurden jedoch in den folgenden Fällen nicht eingehalten:

Modul „ftp“:

Python besitzt mit dem Modul „ftplib“ einen bereits integrierten FTP-Client. Bei dessen Verwendung wäre es jedoch nötig gewesen, die Sicherungen zuerst in einem temporären Ordner zu speichern und danach mittels „ftplib“ auf den FTP-Server zu übertragen.

Ein Grund für die Verwendung eines FTP-Servers zur Sicherung könnte jedoch sein, dass auf der Festplatte des Systems mit der zu sichernden Webanwendung nichtmehr genügend Speicherplatz vorhanden ist. Daher stellt das Mounten des entsprechenden FTP-Verzeichnisses zum direkten Speichern der Sicherungen mittels „curlftpfs“ eine bessere und problemlosere Alternative dar. Leider wurde hierdurch die Plattformunabhängigkeit aufgegeben, da „curlftpfs“ ein Linux-spezifisches Programm ist. Um die Plattformunabhängigkeit von Pywebback wiederherzustellen müssten vergleichbare Funktionen auch für andere Betriebssysteme implementiert werden, was dank des vollständig modularen Aufbaus komfortabel realisierbar wäre.

Modul „backupdb“:

Die Funktion „backupdb“ im gleichnamigen Modul verwendet zur Sicherung der Datenbank das Tool „mysqldump“. Eine Python-Alternative zum Datenbankzugriff wäre hier das Modul „MySQLdb“, allerdings muss dieses nachträglich installiert werden und die Sicherung der gesamten Datenbank wäre hier nicht ganz so komfortabel mit einem einzeiligen Befehl realisierbar. Es müssten z.B. mehrere SELECT Befehle ausgeführt und die Ergebnisse mit weiteren Python Funktionen in eine Datei geschrieben werden.

Die Funktion „mysqldump“ ist dagegen Bestandteil des MySQL-Servers und daher auf jeden Fall vorhanden, sollte eine Datenbanksicherung benötigt werden.

5. Pywiki_install

5.1 Grundlagen

Pywiki_install ist ein Script zur automatisierten Installation von Mediawiki, des Content-Management-Systems der Mediawiki-Foundation. Prominentestes Einsatzbeispiel von Mediawiki ist die Wikipedia.

Zur Auswahl für eine automatisch zu Installierende Webanwendung standen prinzipiell nur Open-Source Produkte. Nur diese bieten eine umfangreiche Einsicht in den Quellcode, welche zwingend notwendig ist, sollte keine zusätzliche Schnittstelle für den Zweck der Automatisierung des Installationsvorgangs vorhanden sein.

In die engere Auswahl haben es letztendlich Mediawiki und Typo3 geschafft. Typo3, weil es das wahrscheinlich am weitesten verbreitete Open-Source Content-Management-System für universelle Einsatzzwecke ist. Mediawiki, weil es überall dort Anwendung findet wo Wissen zu verschiedensten Bereichen zur gemeinsamen Bearbeitung und Betrachtung abgelegt werden soll, insbesondere auch im Hochschulbereich (seit Sommersemester 2011 hat z.B. auch der MI-Studiengang ein eigenes Wiki).

Dabei ist der größte Vorteil von Mediawiki, das es fast jeder kennt, denn fast jeder der Zugang zum Web hat, kenn die Wikipedia.

Sowohl Mediawiki als auch Typo3 sind in den Standard-Paket-Repositories vieler Linux Distributionen enthalten, daher scheinen Scripts zur automatisierten Installation beider Produkte auf den ersten Blick wenig sinnvoll.

Bei Typo3 ist diese Einschätzung auch weitestgehend korrekt. Während der Installation von Typo3 z.B. mit apt-get werden verschiedene Konfigurationsparameter abgefragt, unter anderem für den Datenbankzugriff, sowie eine grundlegende Installation durchgeführt.

Bei Mediawiki wird durch den Aufruf von „apt-get install mediawiki“, jedoch nur eine veraltete Version der Mediawiki heruntergeladen und entpackt. Die eigentliche Installation wird manuell über das Webinterface durchgeführt, also genau auf dieselbe Art und Weise, als würde man die Mediawiki manuell von der offiziellen Website herunterladen und entpacken. Daher wurde die Mediawiki als Grundlage für ein Script zur automatisierten Installation einer Webanwendung ausgewählt.

5.1.1 Grundlegender Aufbau

Das Script zur automatischen Installation der Mediawiki soll im Endeffekt eine Portierung des eigentlichen Mediawiki-Installers, unter Berücksichtigung der in [Kapitel 3](#) definierten Richtlinien darstellen.

Dabei wurde auf Folgende Art- und Weise Vorgegangen:

1) Ansatz – Analyse des Original-Installers „Installer.php“ an Hand des Quellcodes

Dieser Ansatz hat sich auf Grund des enormen Umfangs (ca. 2.300 Zeilen + eingebundene extern implementierte Funktionen) von „Installer.php“ als sehr mühselig, zeitraubend und wenig erfolgsversprechend herausgestellt. Es konnten nur wenige fundierte Erkenntnisse über den Installer gewonnen werden, z.B. das der Installer zwei SQL-Dateien „interwiki.sql“ (Links zu Wikis vieler bekannter Projekte) und „tables.sql“ (Datenbankschema der Mediawiki) aus dem Verzeichnis „maintenance“ in die Datenbank lädt.

2) Ansatz – Manuelle Installation der Mediawiki und Analyse was dabei gemacht wurde

Der originale Mediawiki Installer macht vereinfacht ausgedrückt drei Dinge:

- An Hand der im Webinterface vor der Installation eingetragenen Parameter die Datei „LocalSettings.php“ erstellen. Man wird beim ersten Aufruf aufgefordert diese ins Rootverzeichnis der Mediawiki zu kopieren.
- Datenbankschema anlegen.
- Verschiedene Daten in das zuvor angelegte Datenbank-Schema schreiben (z.B. Admin-User und Passwort, Willkommenstext, Datum).

Die „LocalSettings.php“ wird als einfacher String in der „Installer.php“ angelegt und von dort aus in eine Textdatei geschrieben. Über die Realisierung dieses Vorgangs in „Pywiki_Install“ wird in der entsprechenden Beschreibung des Moduls „LocalSettings.py“ ([Punkt 5.2.4](#)) näher eingegangen.

Um zu analysieren welche Daten der Mediawiki-Installer während der Installation in die Datenbank schreibt, wurde direkt nach der manuellen Installation ein vollständiger SQL-Dump der Datenbank erstellt.

An Hand dieses Dumps wurde dann festgestellt ob überhaupt und welche Daten in die einzelnen Tabellen geschrieben wurden, sowie welche Funktionen dafür verwendet wurden:

Tabellenname	Zeile in db-dump	Inhalt	Zeile in phpfiles
interwiki	340	/maintenance/interwiki.sql	
page	650		installer.php 1331, /includes/Article.php 1569, 1611
revision	956	Installationsdatum	
site_stats	1013	siehe installer.php	installer.php 1278
text	1090	mediawiki wurde erfolgreich installiert...	
user	1203	wikisysop user daten	Installer.php 1309, /includes/User.php 2627
user_groups	1228	funktion addgroup	Installer.php 1319, /includes/User.php 2159

Zuerst wurde das Modul „dbsetup.py“ implementiert, mit grundlegenden Funktionen zum Aufbau einer Datenbankverbindung, zum Laden von SQL-Dateien in die Datenbank und zum Ausführen von SQL-Befehlen.

Danach wurden nach und nach die verschiedenen Funktionen zum Generieren der in die Datenbank eingefügten Werte implementiert, z.B. im Modul „mwikiHash.py“ unter anderem die Funktion zum generieren des Passwort-Hash für den standardmäßigen Datenbankuser.

Um den Komfort zu erhöhen und die Installation der Mediawiki maximal zu automatisieren, wurde zusätzlich zur eigentlichen Mediawiki Installation noch das Modul „Download.py“ implementiert, welches ein beliebiges Tar-Archiv (in diesem Fall die Mediawiki) herunterlädt und entpackt.

Die grundlegende Funktionsweise aller Module und des Hauptscripts wird im nachfolgenden Abschnitt beschrieben.

5.2 Implementierung der Module

Es ist ratsam beim Lesen des folgenden Abschnittes gleichzeitig den Quellcode der jeweiligen Module zu öffnen, um die nachfolgenden Ausführungen besser verstehen zu können.

5.2.1 Modul „configimport“

Das Modul „configimport“ wird bereits unter [Punkt 4.2.2](#) ausführlich beschrieben. Hier kommt der Vorteil einer modularisierten Softwareentwicklung zum Tragen. „Configimport“ konnte für denselben Zweck (Importieren einer Konfigurationsdatei) wie schon in Pywebback unverändert in Pywiki_install übernommen werden.

5.2.2 Modul „dbsetup“

Das Modul „dbsetup“ enthält verschiedene Funktionen zum Zugriff auf MySQL-Datenbanken. Der Original Mediawiki-Installer unterstützt eine Vielzahl an gängigen Datenbanken, unter anderem MySQL, Postgre-SQL, Oracle und DB2. All diese zu implementieren würde den zeitlichen Rahmen des Projektes sprengen und keinen zusätzlichen Vorteil hinsichtlich der Automatisierung einer Mediawiki Installation bringen. Daher soll „dbsetup“ nur den Zugriff auf MySQL-Datenbanken bereitstellen, den wohl gängigsten Datenbank-Typ für Webanwendungen im privaten bis semi-professionellen Bereich.

Direkt aus Python heraus gibt es keine Möglichkeit für den Datenbankzugriff, jedoch wird unter <http://www.python.org/dev/peps/pep-0249/> eine Datenbank-Schnittstelle für Python spezifiziert welche von verschiedenen externen Modulen implementiert werden kann.

Für den Zugriff auf MySQL-Datenbanken implementiert das Modul „MySQLdb“ die Python Datenbank-Schnittstelle, welches wiederum auf das Modul „_mysql“, einen Wrapper für die MySQL-C-API zurückgreift.

„MySQLdb“ lässt sich z.B. unter Ubuntu über das Paket „python-mysqldb“ installieren und ohne weitere Konfiguration in Python Programme einbinden.

Die Funktionen „dbConnect“ und „dbDisconnect“ bauen an Hand der übergebenen Parameter die Verbindung zu einem MySQL-Server oder zu einer spezifischer Datenbank auf selbigen auf und wieder ab. Dabei liefert „dbConnect“ ein Objekt zurück welches die Verbindung zur Datenbank repräsentiert.

Die restlichen Funktionen „executeSQLFile“, „insertSQL“, „executeSQL“, sowie „createDB“ erklären sich durch den Namen bzw. durch den Quelltext und die beigefügten Kommentare von selbst.

5.2.3 Modul „Download“

Die Funktion „wikiExtract“ lädt, sofern in der Konfigurationsdatei erwünscht, von der übergebenen url das Mediawiki-Archiv herunter, und entpackt selbiges in das übergebene Verzeichnis.

Ist der Download-Parameter in der Konfigurationsdatei mit „yes“ definiert, wird die übergebene Variable „url“ als http-url interpretiert, mittels der Funktion „urllib.urlretrieve“ in ein temporäres Verzeichnis heruntergeladen und mittels der Funktion „tarfile.open“ für das nachfolgende Entpacken geöffnet.

Andernfalls wird die Variable „url“ als lokales Verzeichnis interpretiert und direkt mittels „tarfile.open“ geöffnet. Danach wird der Inhalt des Mediawiki-Archivs in das temporäre Verzeichnis entpackt und von dort aus in das Verzeichnis des Webservers kopiert.

Der Umweg über ein temporäres Verzeichnis ist nötig, da die eigentlichen Dateien der Mediawiki innerhalb des Archivs in einem zusätzlichen Ordner mit Namen der aktuellen Mediawiki-Version z.B. „mediawiki-1.16.5“ untergebracht sind. Leider bietet das Python-interne Modul „tarfile“ nicht ohne größere Umwege die Möglichkeit alle Unterordner aus „mediawiki-1.16.5“ inklusive der enthaltenen Dateien zu entpacken.

Nach dem Entpacken werden alle zuvor im temporären Verzeichnis angelegten Ordner und Dateien gelöscht.

5.2.4 Modul „LocalSettings“

Die Funktion „createLocalSettings“ des Modules „LocalSettings“ generiert die für die standesgemäße Funktionalität der Mediawiki benötigte Konfigurationsdatei „LocalSettings.php“ im root-Verzeichnis der Mediawiki.

Hierzu wurden folgende Ansätze probiert:

1) Mitliefern einer fertigen „LocalSettings.php“ die vor der Installation manuell angepasst wird

Der größte Nachteil der sich hierbei ergibt ist, dass vor der Installation zwei Konfigurationsdateien angepasst werden müssen. Man könnte zwar die zusätzlich zur „LocalSettings.php“ benötigten Parameter aus der Konfigurationsdatei des Scripts wie z.B. Datenbankadministrator oder Benutzername und Passwort des Mediawiki-Administrators in die „LocalSettings.php“ integrieren, allerdings müsste man diese dann auch nachträglich wieder entfernen. Eine solche Funktion zum direkten Entfernen oder Ersetzen von Zeilen in Textdateien ist in Python nicht vorhanden und müsste daher extra implementiert werden. Außerdem ist die „LocalSettings.php“ nach PHP-Syntax aufgebaut, was dem Python-internen Config-Parser Probleme bereitet. Daher wurde dieser Ansatz nur für anfängliche Tests verwendet.

2) Mitliefern einer fertigen „LocalSettings.php“ in der alle nötigen Parameter aus der „config.ini“ zur Laufzeit eingetragen werden

Die betroffenen Parameter wurden dabei zuerst manuell aus der mitgelieferten „LocalSettings.php“ entfernt und über das Hauptsript in Kombination mit den Parametern aus der Konfigurationsdatei des Scripts („config.ini“) wieder in die „LocalSettings.php“ geschrieben.

Ein Teil des für diesen Ansatz verwendeten Codes ist noch am Ende des Moduls „LocalSettings“ zu sehen. Der Einfachheit halber wurde auch hier auf das Einfügen der Parameter an ihren ursprünglichen Platz verzichtet und die Parameter einfach an das Ende von „LocalSettings.php“ geschrieben.

Dieser Ansatz erwies sich jedoch als höchst problematisch da die PHP-Variable „\$wgSitename“ in anderen, weiter oben stehenden Variablen der „LocalSettings.php“ referenziert wird. Die Mediawiki war daher mit dieser „LocalSettings.php“ nicht funktionsfähig, im Browser war nur eine leere Seite zu sehen.

3) „LocalSettings.php“ wie in originalem Installer als kompletten String erstellen und in Datei schreiben

Dieser Ansatz erwies sich letztendlich als der sinnvollste. Es sind zwar wie im Quelltext des Moduls „LocalSettings“ zu sehen ist eine Vielzahl an String Operationen nötig, diese lassen sich jedoch sofern einmal angelegt so oft wie benötigt kopieren und anpassen. Einige Parameter werden hierbei zwar fest definiert, lassen sich bei Bedarf jedoch ohne weiteres am Beispiel der anderen Parameter aus der Konfigurationsdatei des Scripts ebenfalls im String eintragen.

5.2.5 Modul „misc_functions“

Das Modul „misc_functions“ ist für Funktionen vorgesehen die nicht in eines der anderen Module passen und für ein eigenes Modul zu trivial sind.

Die enthaltenen Funktionen „wikiTime“ und „wikiWfRandom“ erklären sich bei Ansicht des Quelltextes von selbst und bedürfen daher an dieser Stelle keiner genaueren Erklärung.

5.2.6 Modul „mwikiHash“

Alle Funktionen in „mwikiHash“ wurden ausgehend von den entsprechenden PHP-Funktionen portiert. Die Funktion „mwikiPwHash“ generiert den Passwort-Hash für den Standard-Administrator der Mediawiki, an Hand der Originalen PHP-Funktion.

Die Funktion „mwikiMtoken“ erzeugt ein Token, das gemeinsam mit dem Standard-Administrator in die Tabelle „user“ eingefügt wird. Dieses Token wird ebenfalls in das Cookie eingetragen welches erstellt wird, sollte der Administrator die Funktion zum Erinnern seiner Logindaten beim Anmelden aktivieren.

Über die Funktion „mwikiSecretKey“ wird ein zufälliger SecretKey generiert. Dieser wird zum Erstellen des zuvor beschriebenen Token, sowie fälschungssicherer Cookies verwendet und in die „LocalSettings.php“ eingetragen.

5.3 Implementierung des Hauptscripts

Pywiki_install legt Analog zu Pywebback eine Logdatei an und importiert mittels der Funktion „confimport“ die Konfigurationsdatei. Als nächstes werden verschiedene, im späteren Verlauf benötigte Variablen initialisiert und mit Werten belegt.

Zu beachten sind insbesondere die Listen (Python-Datentyp „List“) zum Einfügen der jeweiligen Werte in die entsprechenden Tabellen. Für jede Tabelle, in die Werte eingefügt werden müssen gibt es zwei Variablen, jeweils für die Attribute und die einzufügenden Werte. Es wäre zwar auf den ersten Blick einfacher gewesen nur die einzufügenden Werte, ohne die entsprechenden Spaltennamen zu definieren (im SQL-Insert Befehl problemlos möglich), jedoch hätte die Verständlichkeit des Codes darunter gelitten.

So kann man exakt herauslesen welcher Wert, in welche Spalte eingefügt wird und welche Bedeutung der jeweilige Wert eventuell hat.

Nach den Variablendeklarationen startet das eigentliche Script. Dieses lädt bei Bedarf zuerst die Mediawiki herunter, entpackt diese und führt mit Hilfe der eingebundenen Funktionen die eigentliche Installation durch.

Der Ablauf des Hauptscripts lässt sich durch Ansicht des Quellcodes, sowie der beigefügten Kommentare und Fehlermeldungen besser verstehen, als durch jede ausführliche textuelle Erklärung. Daher wird an dieser Stelle auf eine noch ausführlichere Erklärung des Hauptscripts verzichtet.

5.4 Fazit

5.4.1 Security-Aspekte

Die Sicherheit von Pywiki_install hängt hauptsächlich von der Sicherheit des ursprünglichen Mediawiki-Installers ab und ist daher, anders als in Pywebback nicht direkt in Pro und Contra Aspekte einzuteilen.

Wie auch schon Pywebback benötigt Pywiki_install nur die minimal für die Installation notwendigen Verzeichnisrechte. Dies sind Schreibrechte auf das Verzeichnis der Logdatei, sowie auf das Verzeichnis in welches die Mediawiki-Dateien entpackt werden sollen (sofern diese nicht bereits dort vorhanden sind).

Sicherheitsprobleme sind unter anderem das wie schon in Pywebback alle Passwörter in der Konfigurationsdatei im Klartext gespeichert werden, was hier jedoch anders als im Falle von Pywebback von geringerer Bedeutung ist. Die Installation wird in der Regel pro System nur einmal durchgeführt, daher kann die Konfigurationsdatei nach einer erfolgreichen Installation gelöscht werden. Für diesen Zweck wurde zum Ende des Hauptscripts bereits der passende Code implementiert, welcher bei Bedarf nur noch auskommentiert werden muss.

Ein weiteres Problem ist, dass in der „LocalSettings.php“ das Passwort für den Datenbankbenutzer ebenfalls in Klartext gespeichert ist. Aber auch dies ist ein spezifisches Problem der Mediawiki und kann daher in Pywiki_install nicht berücksichtigt werden.

5.4.2 Mögliche Einsatzbereiche

Generell lässt sich Pywiki_install überall einsetzen wo die Mediawiki installiert werden soll.

Ein besonders sinnvolles Anwendungsszenario, wäre jedoch bei einem Webhoster, bei dem immer wieder Mediawiki-Installationen stattfinden. Der Vorteil von Pywiki_install gegenüber z.B. einer imagebasierten Lösung wäre hier, dass die Installation an Hand der Konfigurationsdatei problemlos an die Bedürfnisse des jeweiligen Nutzers bzw. Kunden angepasst werden kann. Auch wäre eine rein imagebasierte Lösung unsicher, da überall dieselben Passwörter eingetragen wären, welche der Nutzer nachträglich ändern müsste.

Bei Pywiki_install mit einer Konfigurationsdatei ohne eingetragene Standard-Passwörter, ist der Nutzer dazu gezwungen eigene Passwörter zu wählen oder man könnte alternativ zufällige Passwörter generieren. Dabei wäre es im Gegensatz zur imagebasierten Lösung nahezu ausgeschlossen, dass mehrere Mediawiki-Installationen der Kunden des Webhosters dieselben Passwörter benutzen.

5.4.3 Allgemeines Fazit

Pywiki_install hat alle zuvor definierten Ziele erreicht, auch die Richtlinien zur Script-Entwicklung aus [Kapitel 3](#) wurden vollständig umgesetzt. Es ist schneller, weniger komplex und übersichtlicher als der Original-Installer und vollständig plattformunabhängig.

Pywiki_install wurde an Hand der Mediawiki Version 1.16.5 implementiert, jedoch war nach einem kurzen Test auch die Installation der aktuellen Mediawiki Version 1.17 problemlos möglich. Dort haben sich unter anderem das Datenbankschema und die Syntax der „LocalSettings.php“ verändert. Im Falle des Datenbankschemas waren keinerlei Änderungen nötig, da dieses in der Datei „tables.sql“ definiert wird, welche ohnehin von „Pywiki_install“ aufgerufen wird (siehe unter anderem Modul „dbsetup“ [Punkt 5.2.2](#)). Die Syntax der „LocalSettings.php“ sowie alle anderen aktuellen und zukünftigen Änderungen an der Mediawiki, lassen sich auf Grund des modularen Aufbaus einfach und ohne wesentliche Änderungen am Hauptscript auf Pywiki_install übertragen.

Aktuell noch vorhandene Mängel von Pywiki_install sind, das im Gegensatz zum originalen Installer nur MySQL-Datenbanken unterstützt werden und z.B. der Willkommenstext noch fest auf Deutsch eingetragen ist.

6. Abschluss

6.1 Erfüllung der Ziele

Hier soll erläutert werden in welchem Umfang und ob überhaupt die unter [Punkt 1.2](#) definierten Ziele erreicht wurden.

6.1.1 Serverinstallation/Einrichtung

1) Einrichten eines Linux-Servers mit FAI (Fully Automatic Installation) zur automatischen Installation von Linux-Systemen mittels PXE-Boot, mit Auswahl der geeigneten Distributionen, Installation und Konfiguration des Servers.

Der Linux-Server mit FAI ist voll funktionsfähig und installiert wie in der Zielsetzung definiert ein Linux-System (als Beispiellösung-Server Ubuntu 10.10), inklusive individueller Anpassungen ohne menschliche Eingriffe. Jedoch wurden bei der Herangehensweise an diesen Teil des Projektes zahlreiche Fehler gemacht, welche in einem unverhältnismäßig hohen Zeitverbrauch resultierten (Siehe [Punkt 2.3](#)).

2) Erstellen von geeigneten Images für die automatisch zu installierenden Web-Server.

FAI ist vom Grundprinzip her nicht imagebasiert, die Pakete werden wie bei einer normalen Linux-Installation mit dem Paketmanagement-Tool der jeweils zu installierenden Distribution definiert. Die zu installierende Distribution wird nicht über zentrale Images, sondern an mehreren Stellen im sogenannten Configspace definiert (siehe Kapitel 2).

3) Programmieren eines Python-Scripts zur automatischen Installation und Einrichtung einer LAMP-Umgebung (unter Verwendung von Paket-Management-Tools der jeweiligen Distribution).

Dieses Ziel erwies sich als Überflüssig, da eine LAMP-Umgebung über einfache einzeilige Aufrufe des Paketmanagement-Tools der jeweiligen Distribution installiert werden kann, z.B. „apt-get install lamp-server^“. Eine Python-Abstraktion dieses Vorgangs schien unlogisch und hätte nur unnötig Zeit gekostet. Bei der Verwendung von FAI lässt sich eine LAMP-Umgebung ebenfalls unter Angabe der Paketnamen, in der entsprechenden Konfigurationsdatei automatisiert installieren.

6.1.2 Installation der Web-Applikationen

1) Programmieren mehrerer Python-Scripts zur automatischen Installation und Einrichtung verschiedener Web-Applikationen (z.B. für die fünf mit dem größten Verbreitungsgrad). Eventuell auch einzelnes Script für alle Web-Applikationen.

Dieser Teil des Projekts wurde durch Pywiki_install (siehe Kapitel 4) zumindest Abschnittsweise erfüllt. Die Zielsetzung gleich fünf Scripts zur automatischen Installation von Webanwendungen erwies sich als zu Ehrgeizig. Die Umsetzung des originalen PHP-basierten Installers der Mediawiki, in ein Python-Script, ohne vorherige Kenntnisse des Quellcodes und vor allem ohne PHP-Kenntnisse erwies sich als mühselig und zeitraubend.

Außerdem hat zum Zeitpunkt des Beginns der Entwicklung an den Python-Scripts, die Einrichtung der FAI-Lösung bereits annähernd die Hälfte des Semesters und der Projektzeit in Anspruch genommen.

Die eigentliche Entwicklung der Scripts lief jedoch auf Grund der zuvor definierten Richtlinien (siehe [Kapitel 3](#)) wesentlich effektiver ab, als zuvor die Einrichtung der FAI-Lösung.

2) Erstellen eines anpassbaren Muster-Python-Scripts als Vorgabe für zusätzliche Web-Applikationen die nicht bereits im Projekt enthalten sind.

Dieses Ziel schien bereits nach den Anfängen der Umsetzung von Pywiki_install sehr unrealistisch. Die PHP-basierten Installer der jeweiligen Webanwendungen sind sehr umfangreich und haben bis auf die gemeinsame Programmiersprache PHP, sowie die Verwendung eines webbasierten Formulars zur Eingabe der Installationsparameter wenig gemeinsam. Ein Muster Script als Allgemeine Lösung für die automatisierte Installation von Webanwendungen erscheint daher relativ unrealistisch.

Trotzdem wurde der eigentliche Gedanke hinter diesem Ziel, nämlich die Erleichterung der automatisierten Installation von Webanwendungen mittels Python, für andere Entwickler teilweise erreicht. Das Modul „dbsetup“ ([Punkt 5.2.2](#)) lässt sich z.B. völlig unabhängig von Pywiki_install auch für den Datenbankzugriff bei der Installation von beliebigen anderen Webanwendungen mittels eines Python Scripts verwenden. Auch das Modul „Download“ ([Punkt 5.2.3](#)) wäre für die Installation anderer Webanwendungen brauchbar.

Generell ist es auch für andere Entwickler mit ähnlichen Zielsetzungen empfehlenswert, auf die unter [Punkt 4.1.1](#) beschriebene Vorgehensweise zur Umsetzung des Installers einer Webanwendung in eine andere Script-basierte Programmiersprache zurückzugreifen.

3) Integration von rudimentären Administrations-Aufgaben für die Web-Applikationen durch das jeweilige Script (z.B. automatische Sicherheitsupdates, Erstellen von Cronjobs durch das Script für eine zeitgesteuerte Sicherung von Files und Datenbanken).

Hier war die Zielsetzung wieder zu ehrgeizig und es konnte mit Pywebback nur eine, jedoch essenziell wichtige Administrationsaufgabe automatisiert werden. Auch wäre es unpassend gewesen die jeweils zu automatisierende Administrationsaufgabe in das Installations-Script zu implementieren. Das Installations-Script wird in der Regel nur einmal aufgerufen, das entsprechende Script zur Administrations jedoch regelmäßig. Daher macht eine zusätzliche Stufe der Modularisierung (Trennung von Installation und Administration) wie sie mit Pywebback realisiert wurde wesentlich mehr Sinn, als die ursprüngliche Zielsetzung.

4) Anwender legt grundlegende Parameter für die Installation der Webanwendungen in Konfigurationsdateien mit vordefinierten Parametern fest.

Dieses Ziel wurde mit den Konfigurationsdateien von Pywebback und Pywiki_install, in Kombination mit dem Modul „Configimport“ (Siehe [Punkt 4.2.2](#)) vollständig umgesetzt.

5) Scripts sollen auch unabhängig von der FAI-Lösung auf bereits installierten Servern lauffähig sein (Vollständige Trennung von Serverinstallation und Installation der Web-Applikationen, Anpassung an Paket-Management Tools verschiedener Distributionen).

Dieses Ziel wurde ebenfalls vollständig umgesetzt. Eine Anpassung an verschiedene Paket-Management Tools erwies sich als überflüssig, da diese für Pywiki_install nicht benötigt werden. Jedoch wurde die mögliche Integration in die FAI-Lösung (auch aus Zeitmangel) vernachlässigt und kein Test durchgeführt ob Pywiki_install auch während einer vom FAI-Server aus durchgeführten Installation funktioniert. Dies sollte allerdings auch ohne Test, auf Grund der vielseitigen Konfigurations- und Anpassungsmöglichkeiten im Configspace von FAI problemlos möglich sein.

5.2 Abschließendes Fazit

Im gesamten Kontext gesehen war dieses Projekt ein voller Erfolg. Wie im vorherigen Abschnitt zu sehen wurden alle Ziele entweder vollständig oder teilweise erfüllt. Keines der Ziele ist wirklich fehlgeschlagen. Dennoch wurden im Verlauf des Projektes einige Fehler begangen. Eine ausreichende Planung wurde erst zu Beginn der Entwicklung der Scripts durchgeführt. Projektmanagement fand nur stark eingeschränkt und in unregelmäßigen Abschnitten unter Zuhilfenahme des Ticket-basierten Web-Projektmanagementsystems Redmine statt. Neben den bereits erwähnten Zielen wurden im Rahmen dieses Projekts folgende Lerneffekte erzielt:

- Intensive Einarbeitung in Linux, insbesondere in einer reinen Shell-Umgebung.
- Umfangreiche Einarbeitung und Kenntnisse über FAI.
- Programmiersprache Python im Allgemeinen, sowie viele Module, Konzepte und Eigenarten der Sprache gelernt.
- Planung bzw. Projektmanagement sollten im Rahmen eines Projekts niemals vernachlässigt werden.
- Bei einem Problem ist es oftmals sinnvoller sich intensiv mit einer Dokumentation auseinander zu setzen, anstatt einfach nur nach einer möglichen Lösung zu googeln.

7. Anhang

7.1 Glossar

Hier sollen einige Begriffe erklärt werden, die entweder eine gewisse Bedeutung für die Funktion der jeweiligen Bestandteile des Projekts haben oder einfach nur häufig erwähnt wurden.

Curlftpfs - <http://curlftpfs.sourceforge.net/>

Linux-Tool auf Basis von FUSE um FTP-Server in den Userspace zu mounten.

DHCP - <http://de.wikipedia.org/wiki/DHCP>

Netzwerkprotokoll zur automatischen Konfiguration von Netzwerkparametern.

GRUB - <http://www.gnu.org/software/grub/>

Standard Bootloader der meisten gängigen Linux-Distributionen. Aktuell in der Version 2.

Hash - <http://de.wikipedia.org/wiki/Hashfunktion>

Verschiedene Bedeutungen. Im Kontext dieses Projekts Algorithmus zur Verschlüsselung von Passwörtern.

Iptables - <http://www.netfilter.org/projects/iptables/index.html>

Shell Tool zur Konfiguration des im Linux-Kernel integrierten Netfilter Frameworks. Wird zum Filtern und Umleiten (Firewall) von IP-Paketen verwendet.

LAMP - <http://de.wikipedia.org/wiki/LAMP>

Linux, Apache, MySQL, PHP. Weit verbreitete Umgebung für Webanwendungen.

MySQL - <http://www.mysql.de/>

Weit verbreitete Open-Source Datenbank von Oracle (ursprünglich Sun).

NAT - http://de.wikipedia.org/wiki/Network_Address_Translation

Setzt bestimmte IP-Adressen und Ports in andere um. Wird insbesondere dazu verwendet um Netze mit privaten IP-Adressen (z.B. LAN) mit Netzen die öffentliche IP-Adressen verwenden (z.B. Internet) zu verbinden.

NFS - <http://nfs.sourceforge.net/>

Network File System: Protokoll zum netzwerkbasierten Zugriff auf Dateien unter Unixoiden Betriebssystemen.

PXE - http://de.wikipedia.org/wiki/Preboot_Execution_Environment

Verfahren um beliebige Computerhardware über Netzwerk zu booten. Wird von nahezu allen Ethernet-basierten Netzwerkinterfaces unterstützt (auch in virtuellen Maschinen).

Tar - <http://www.gnu.org/software/tar/>

GNU- Programm zum Erstellen und entpacken von .tar Archiven.

Tftp - http://de.wikipedia.org/wiki/Trivial_File_Transfer_Protocol

Vereinfachte Variante des ftp-Protokolls. Unterstützt ausschließlich das Lesen und Schreiben von Dateien und wird häufig in Verbindung mit PXE verwendet.

VLAN - <http://de.wikipedia.org/wiki/VLAN>

Logische Einteilung von Netzen auf Ebene 2 des ISO/OSI-Modells.

7.2 Linkverzeichnis

7.2.1 FAI

<http://fai-project.org/>

Offizielle Website des FAI-Projekts, mit Dokumentationen, FAQs, Manpages etc.

<http://wiki.fai-project.org>

Offizielle Wiki des FAI-Projekts

<http://www.akoestica.be/blog/home/sysadmin/15-provisioning/49-installing-fai>

Grundlegende Installationsanleitung für FAI

http://wiki.fai-project.org/wiki/UbuntuJauntyInstallationHowTo#Setting_up_FAI

Installation von Ubuntu 9.04 mittels FAI

7.2.2 Scripts

<http://python.org/>

Offizielle Python Website

<http://diveintopython.org/>

Frei verfügbares Buch zur Einarbeitung in Python

<http://mysql-python.sourceforge.net/MySQLdb.html>

MySQLdb – Python Modul für den Zugriff auf MySQL-Datenbanken

<http://www.eclipse.org/>

Eclipse – Allgemeine Bekannte Open-Source Entwicklungsumgebung

<http://pydev.org/>

PyDev – Eclipse Plugin für Python-Entwicklung

<http://subclipse.tigris.org/>

Subclipse – SVN-Connector für Eclipse

<http://phpxref.com/xref/mediawiki/functions/index.html>

Übersicht über Referenzen der PHP-Funktionen in Mediawiki