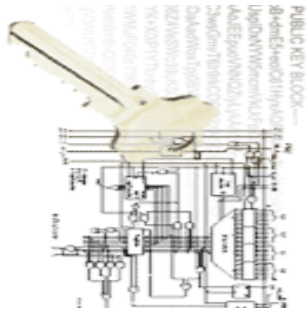
Graz University of Technology
IAIK
Institute for Applied Information Processing and Communications
Inffeldgasse 16a
A–8010 GRAZ

IIS Homework



Securing Linux

Version 1.0

Thomas Andergassen 0130490,
Hartwig De Colle 0130513,
Markus Köberl 0130163, and
Susanne Schöberl 0130374

Article
IIS - T 6.2

http://www.iaik.tu-graz.ac.at/teaching/

Securing Linux

Thomas Andergassen, Hartwig De Colle, Markus Köberl, and Susanne Schöberl

31. Januar 2008

Zusammenfassung

Sicherheit ist in der IT ein sehr wichtiger Faktor. Zu einem sicheren System gehört sowohl Schutz vor lokalen als auch vor entfernten Angriffen. Dieses Dokument gibt einen kurzen Überblick über die in Linux integrierten Möglichkeiten das System vor solchen Angriffen zu schützen. Weiters werden einige moderne Erweiterungen vorgestellt um diese Möglichkeiten noch zu erweitern.
Keywords: Linux, Security, Rights, PAM, Firewall, SE-Linux, RSBAC

Inhaltsverzeichnis

1 Motivation

Sicherheit ist ein wichtiges Thema für jeden Computerbenutzer, bzw sollte es sein. Der Schutz der eigenen Daten oder der Schutz vor dem Missbrauch des eigenen Systems sollte in der Zeit der Cracker, Viren, Würmer, Spammer, Phisher und Trojaner eine wichtige Rolle im Alltag eines Pc-Users und vor allem eines Admins spielen. Nicht nur der Verlust oder die Kompromitierung der sensiblen Daten ist eine Gefahr, auch der Missbrauch des eigenen Systems z.B. zum Versenden von Spam oder für Angriffe auf andere Systeme können großen Ärger bringen. Wer allerdings glaubt das betrifft nur Produkte aus Redmond ist auf dem Holzweg. Ein ungenügend gesichertes Linux-System ist genauso angreifbar. Die gute Nachricht ist allerdings, dass Linux von Haus aus sehr gute Möglichkeiten bietet um ein System abzusichern.

2 Sicherheit, die Basics

2.1 Von Benutzern und Gruppen

Die Grundlage der Mechanismen spielen Benutzer und Gruppen mit ihren Rechten. Jeder der mit dem Pc arbeiten will, wird durch einen Benutzer repräsentiert der durch ein Passwort geschützt wird. Aber nicht nur Personen die das System benutzen, haben ein Benutzerkonto, sondern auch Daemonen, also Programme die Services auf dem Rechner anbieten. Eine Gruppe dient dazu mehrer Benutzern die gleichen Rechte zu geben und erleichtert dadurch die Verwaltung. Jeder Benutzer kann einer oder mehreren Gruppen angehören. Es gibt einen speziellen Benutzer, den Systemverwalter bzw `root` der alles darf. Es soll nicht mit `root` Rechten gearbeitet werden da so auch Schadsoftware mit den Selben ausgeführt wird.

2.1.1 Zugriffsberechtigungen

Arbeiten mehrere Benutzer an einem Rechner so muss gewährleistet werden, dass sie sich nicht in die Quere kommen. Das heisst, dass sie sich nicht gegenseitig die Dateien ohne Erlaubnis verändern dürfen. Um dies handzuhaben stellt das Filesystem sogenannte Zugriffsberechtigungen (Dateiattribute) zur Verfügung (siehe [?] und [?]). Auch wird jedem Benutzer ein eigenes `home` Verzeichnis, zugeteilt. In diesem kann er all seine Dateien speichern. Auf die `home`-Verzeichnisse anderer Benutzer kann im Normalfall nicht zugegriffen werden.

Jedem Verzeichnis und jeder Datei liegen drei verschiedene Rechte zu Grunde (siehe [?]):

Lesen: Auf Englisch *read* (abgekürzt mit `r`) ist das Recht eine Datei zu lesen. Bei einem Verzeichnis jedoch bedeutet es, das der Inhalt angesehen werden kann.

Schreiben: Auf Englisch *write* (abgekürzt mit `w`) ist das Recht eine Datei oder ein Verzeichnis anzulegen, zu bearbeiten oder zu löschen.

Ausführen: Auf Englisch *execute* (abgekürzt mit `x`) ist das Recht ausführbare Programme zu starten. Bei einem Verzeichnis wird der Zugriff damit geregelt. Sollte es nicht gesetzt sein, so kann in dieses Verzeichnis nicht gewechselt werden.

Für jedes Verzeichnis und jede Datei werden diese Rechte drei mal vergeben und zwar jeweils für:

Eigentümer: Auf Englisch *owner* ist der Besitzer der Datei oder des Verzeichnisses. Im Normalfall ist der, der es erstellt hat auch der Besitzer. Der Besitzer kann aber auch vom `root` festgelegt werden.

Gruppe: Auf Englisch *group* (abgekürzt mit `g`) ist eine Benutzergruppe. Sollte ein Benutzer Mitglied dieser Gruppe sein und nicht der Besitzer, so bekommt er die Rechte der Gruppe.

Andere: Auf Englisch *others* (abgekürzt mit `o`) gilt für alle anderen, also für jene die nicht Mitglied der Gruppe und auch nicht Besitzer sind.

Mit Hilfe des Befehls `ls -la` in einem Terminal können die Dateiattribute betrachtet werden.

Ein Aufruf könnte wie folgt aussehen:
`-rw-r-r- 1 tux users 1234 Dez 21 12:12 beispiel.txt`
Wobei `tux` der Eigentümer ist und `users` die Gruppe. Die Dateiberechtigungen werden durch `-rw-r-r-` beschrieben. Bei diesem Beispiel (siehe Tabelle 2) darf der Benutzer `tux` in die Datei `beispiel.txt` schreiben und von ihr lesen, die Gruppe `users` in welcher der Benutzer `tux` ist, darf nur lesen auch alle anderen dürfen nur lesen (siehe [**?**]). Als erstes wird jedoch der Typ gekennzeichnet, es gibt verschiedene Typen (siehe 1).

Typ	Abkürzung
Verzeichnis	`d`
Block-Device	`b`
Char-Device	`c`
FIFO/PIPES	`p`
Socket	`s`
Datei	`-`

Tabelle 1: Verschiedene Typen

Typ	Eigentümer			Gruppe			Anderen			Datei
	read	write	execute	read	write	execute	read	write	execute	
-	r	w	-	r	-	-	r	-	-	beispiel.txt

Tabelle 2: Berechtigungen für die Datei beispiel.txt von User `tux` mit der Gruppe `users`

Zusätzlich zu den bereits bekannt Rechten gibt es noch zwei spezielle Berechtigungen. Zum einen *set user ID on execution* kurz SUID, damit kann ein Programm mit den Rechten des Eigentümers gestartet werden. Zum anderen *set group ID on execution* kurz GUID, damit wird ein Programm mit den Rechten der Gruppe gestartet. Diese Beiden können sehr gefährlich werden, denn wenn das Programm `root` gehört, so wird jeder der es ausführt für das Programm zum `root`. Sollte ein Programm unter andern Rechten ausgeführt werden so wird das Programm `sudo` empfohlen.

Wichtig ist jedoch das die Berechtigungen nur soweit vergeben werden wie sie auch gebraucht werden. Das heisst wenn eine Gruppe eine Datei nicht verändern muss, so braucht sie auch nicht das Schreib-Recht zu haben.

2.2 Erweiterung durch Access Control Lists

Manchmal reichen die normalen Berechtigungen nicht mehr aus, dafür gibt es dann die *Access Control Lists* kurz ACL. Jedoch muss das vom Filesystem unterstützt werden z.B. XFS.

Mit den normalen Rechten hat jede Datei bzw. Verzeichnis genau einen Benutzer und eine Gruppe, ACL erlauben beliebig viele Benutzer- und Gruppeneinträge (siehe [**?**]). Mit Hilfe der ACL können auch Standardeinstellungen für neue Dateien oder Verzeichnisse vorgenommen werden. Wird ein neues Verzeichnis in einem Verzeichnis das eine ACL besitzt erstellt so erbt es die Einstellungen von diesem.

Jedoch sollte sehr umsichtig mit den ACL umgegangen werden um sich nicht in den ganzen Berechtigungen zu verlieren. Besonders erwähnenswert ist, das bei den ACL auch mithilfe einer `mask` die maximalen Berechtigungen angegeben werden können. Sollte die maximale Berechtigung nur lesen sein, so kann auch ein Benutzer, dem auch das Schreibrecht gegeben wurde, nur lesen.

2.3 Kontingent oder Quota

Festplatten können alleine durch die Logeinträge von verschiedenen Programmen, lange Laufzeit vorrausgesetzt, voll werden. Auch ein Angriff kann darauf abzielen eine Festplatte zu füllen. Wird die Festplatte zu voll funktioniert das System nicht mehr richtig und ist dadurch angreifbar.

Ein Kontingent oder englisch Quota stellt eine Begrenzung des verfügbaren Speicherplatzes dar, so das sichergestellt werden kann das dem System immer genug Speicherplatz zur Verfügung steht. Quotas darf nur der `root` vergeben (siehe [**?**]). Quotas müssen für jede Partition extra vergeben werden und können per Benutzer oder Gruppe vergeben werden, wobei Benutzer Quotas eine höhere Priorität haben. Ein Benutzer der auf mehreren Partitionen Schreibrechte hat, muss auch auf jede dieser ein Quota haben. Bei den Quotas kann folgendes eingestellt werden:

Softlimit: Die Grenze darf nur kurz überschritten werden und der Benutzer wird extra gewarnt.

Grace Period: Legt den Zeitraum fest in der das Softlimit überschritten werden darf.

Hardlimit: Legt die Grenze fest, die nicht überschritten werden darf.

Um mit Quotas arbeiten zu können muss der Kernel mit der Option `Quota Support` kompiliert werden oder worden sein. Das Filesystem muss ebenfalls Quotas unterstützen. Auch muss die Datei `/etc/fstab` angepasst werden. Im Wurzelverzeichnis der jeweiligen Partition müssen die Dateien `aquota.user` und `aquota.group` erstellt werden. Der Lesezugriff auf diese quota-Dateien ist notwendig, damit die Benutzer auch den aktuellen Stand ihres Kontingentes erfragen können.

2.4 Richtiges Partitionieren

Es sollten sich das System, die Auslagerungsdatei und die Benutzerverzeichnisse auf getrennten Partitionen oder Festplatten befinden, das Erleichtert zum einem das Backup und zum andern erhöht es die Sicherheit. So kann es nicht mehr passieren, das das System aufgrund fehlenden Festplattenspeichers nicht mehr funktioniert.

Auch ist es wichtig das Partitionen richtig gemountet, also in das Filesystem eingehängt werden. Als Beispiel dient folgender Eintrag in die `etc/fstab`
`/dev/hda9 /tmp ext2 defaults,nosuid,noexec,nodev 0 2`
Für eine Beschreibung der Optionen und deren Wirkung siehe Tabelle 3.

Option	Beschreibung
`defaults`	Entspricht den voreingestellten Optionen `rw, suid, dev, exec, auto, nouser,` und `async`.
`rw`	Einhängen des Dateisystems zum Lesen und Schreiben.
`ro`	Einhängen des Dateisystems ausschlieSSlich zum Lesen, Schreiboperationen werden ignoriert.
`user`	Ein Normaluser darf das Dateisystem einhängen. Ansonsten darf dies nur `root`.
`nouser`	Nur `root` darf das Dateisystem einhängen.
`dev`	Das Nutzen von Gerätedateien auf diese Partition ist erlaubt.
`nodev`	Das Nutzen von Gerätedateien auf diese Partition ist nicht erlaubt.
`exec`	Programme auf der Partition können ausgeführt werden.
`noexec`	Programme auf der Partition können nicht ausgeführt werden.
`auto`	Partition wird beim Booten automatisch eingehängt.
`noauto`	Partition wird beim Booten nicht automatisch eingehängt.
`atime`	Die Zugriffszeit wird bei jedem Zugriff gesetzt.
`noatime`	Die Zugriffszeit nicht gesetzt.
`suid`	SUID und SGID Bits werden interpretiert.
`nosuid`	SUID und SGID Bits werden nicht interpretiert.
`sync`	Ein- und Ausgabeoperationen werden synchron durchgeführt.
`async`	Ein- und Ausgabeoperationen werden asynchron durchgeführt.
`usrquota`	Unterstützung für Userquota.
`grpquota`	Unterstützung für Gruppenquota.

Tabelle 3: Auszug aus den verschiedenen Optionen für mount

Zur Sicherheit trägt das Partitionieren in dem Sinne bei, das Parititionen von denen nur gelesen werden soll, so einbindet das man nur lesen kann. Auch kann so verhindert werden das z.b: Progromma von einem USB-Stick gestartet werden können.

2.5 Hinter Gittern, im Jail

Darunter versteht man eine `chroot` Umgebung (`chroot` steht für `change root`). Diese Funktion bewirkt ein verändern des Rootverzeichnisses und wirkt sich nur auf den aktuellen Prozess und die Kinderprozesse aus. Ein Programm das in einer Root Umgebung gestartet

wurde, kann nur mehr auf Files innerhalb des neuen Rootverzeichnisses zugreifen. Nur der root darf `chroot` starten.

Diese Umgebung kann verwendet werden um sicherheitskritische Programme vom übrigen Filesystem zu isolieren. Die Verwendung dieser Umgebung wird dadurch erschwert das Programme bestimmte Daten bei der Ausführung brauchen. Daher müssen alle Daten, die das Programm benötigt, in die `chroot` Umgebung kopiert werden. Jedoch sollte darauf geachtet werden, das nicht mehr als nötig kopiert wird. Sollte in diese Umgebung eingebrochen werden, so kann nur minimaler Schaden angerichtet werden.

Mit Hilfe der `chroot` Umgebung könnte auch der Fernzugriff über SSH sicherer gestalltet werden, in dem jeder SSH User in so einer Umgebung eingesperrt wird, oder der SSH Server selbst in einer läuft. Näheres sieh unter [**?**].

2.6 Verschlüsselung

Alle Daten werden unverschlüsselt auf der Festplatte abgespeichert, mit Ausnahme der Passwörter. Bei hoch sensiblen Daten ist es jedoch besser wenn sie verschlüsselt weden, dadurch können sie auch dann nicht gelesen werden, wenn das Benutzer Passwort in die falschen Hände gerät.

Darüber hinaus ist es mit den Standard Rechten nicht möglich ein Dokument weiterhin zu schützen wenn es den eigenen Computer verlässt, also über das Internet verschickt wird. Da eignet sich eine Verschlüsselung besser. Um etwas zu Verschlüsseln, sei es eine Datei, Verzeichnis oder eine Partition werden zwei Schlüssel gebraucht. Einen öffentlichen und einen privaten Schlüssel.

Wird eine ganze Festplatte verschlüsselt, so kommt man auch dann nicht an die Daten heran, wenn die Festplatte in einem anderen Pc eingebaut wird. Demnach eignet sich eine Verschlüsselung vorallem beim Firmen Laptops um so Daten bei einem Diebstahl zu schützen.

3 Pass. . . , wie war das gleich nochmal?

3.1 Einfache Authentifizierung

Wie schon im Kapitel 2.1 erwähnt, werden die Benutzer über Passwörter authentifiziert. In den Anfängen von Linux wurden alle Benutzer, ihre Gruppe und die Hashes der Passwörter in der Datei `/etc/passwd` gespeichert. Das ist sehr unsicher da die `/etc/passwd` für alle Benutzer lesbar sein muß, damit festgestellt werden kann zu welchen Gruppen ein Benutzer gehört. Damit besteht aber auch für jeden Benutzer die Möglichkeit die Hashes der Passwörter auszulesen und automatisierte Angriffe darauf zu starten. 1992 wurde das Shadow Password System auf Linux portiert. Dabei werden Benutzer und Gruppen weiterhin in `/etc/passwd` gespeichert, ihre Passwörter aber in der geschützten `/etc/shadow`. Diese ist nur vom System auslesbar, nicht von den Benutzern und erlaubt so mehr Sicherheit. Heute ist das PAM System, das Pluggable Authentication Module System, Standard in allen gängigen Linux Distributionen. Es ist, wie der Name schon sagt, ein modulares System in das sich verschiedene Authentifizierungsmechanismen integrieren lassen, unter anderem auch das Shadow- System.

3.2 Wie PAM funktioniert

Der Vorteil von PAM ist, dass die Authentifizierungsmechanismen in Modulen implementiert sind. Bei einem Fehler in einem Modul oder beim Umstieg auf einen anderen Authentifizierungsmechanismus müssen daher die Programme die PAM unterstützen nicht neu kompiliert werden.
Für jedes Programm kann ein eigenes Konfigurationsfile im Verzeichniss `/etc/pam.d` erstellt werden. Dabei ist der Filname gleich dem Programmnamen zu wählen, so existieren normalerweise nach der Installation für die wichtigsten Programme bereits Defaulteinstellungen. So zum Beispiel für `login`, `passwd`, `cron` und `su`. Für alle Programme ohne wird das Konfigurationsfile `other` verwendet.

Konfigurationsformat:
Modul Typ Kontroll Typ Modul Parameter
Mit `@include Filename` kann auch ein anderes Konfigurationsfile inkludiert werden.

Modul Typ: Einer der vier Modul Typen (`auth`, `session`, `account`, `password`).

Kontroll Typ: Einer der vier Kontroll Typen (`requisite`, `required`, `sufficient`, `optional`).

Modul: Pfad und Name des Modules. Der Pfad ist dabei optional, wenn er nicht angegeben wird muss sich das Modul im Verzeichniss `/usr/lib/security` befinden.

Parameter: Argumente die an das Modul direkt übergeben werden. Diese sind für jedes Modul unterschiedlich.

Es gibt vier Typen von Modulen die unterschieliche Aufgaben erfüllen:

auth: Erledigt in der Regel die Passwortabfrage und ist somit für die Authentifizierung zuständig.

session: Regelt das Setzen von Umgebungsvariablen, das Mounten und Unmounten von Laufwerken oder das Protokollieren von Ereignissen.

account: Damit lässt sich der Zugriff auf einen Rechner einschränken auf Basis der Tageszeit, von Systemressourcen oder vom Standort des Benutzers. Die Überprüfung ob ein Account abgelaufen ist erfolgt auch an dieser Stelle.

password: Ist für die Ändernung von Passwörter zuständig.

Die vier verschiedene Kontroll Typen:

requisite: Ein Fehler führt zum sofortigen Abbruch.

required: Bei einem Fehler werden noch weitere Module ausgeführt.

sufficient: Bei Erfolg wird der Zugang erlaubt auch wenn ein vorheriges Modul fehlerhaft war.

optional: Optional, Ergebniss hat keinen Einfluss außer wenn es das einzige Modul ist.

Zum besseren Verständniss ein kleines konfigurations Beispiel:

```
account   required  pam_unix.so
auth      required  pam_unix.so  nullok_secure
password  required  pam_unix.so  nullok obscure min=4 max=8 md5
session   required  pam_unix.so
```

`nullok_secure` bewirkt dass sich User mit leerem Passort einloggen können wenn sie es über eine Konsole versuchen welche im File `/etc/securetty` deffiniert sind. `nullok obscure min=4 max=8 md5` bewirkt dass das neue Passwort nicht leer sein darf, min 4 und max 8 Zeichen haben muss und im md5 Format gespeichert wird.

3.3 Interessante Module

Wartung und Informationen über des Systems:
`pam_nologin` - Verhindert das Login von Benutzern wenn die Datei `/etc/nologin` existiert und gibt den Inhalt dem Benutzer aus.
`pam_mkhomedir` - Erstell das Homeverzeichniss für den Benutzer wenn noch keines vorhanden ist.
`pam_limits` - Zur Limitierung von Resoursen wie zum Beispiel: maximale Anzahl an Logins, maximale verwendete CPU-Zeit, maximaler Speicherverbrauch.
`pam_env` - Zum Setzen und Löschen von Environment-Variablen.
`pam_lastlog` - Gibt Datum und Uhrzeit des letzten erfolgreichen Logins aus.
`pam_motd` - Gibt bei erfolgreichen Login den Inhalt von `/etc/motd` aus (message of the day).

`pam_mail` - Gibt `you have new mail` aus wenn im Mailfolder ein ungelesenes Mail vorhanden ist.

Einige Module für die zentrale Authentifizierung:
`pam_http` - http/https
`pam_krb5` - MIT Kerberos
`pam_ldap` - LDAP
`pam_mysql` - MySQL server
`pam_ncp` - NetWare server
`pam_pgsql` - PostgreSQL database
`pam_radius-auth` - RADIUS
`pam_smbpass` - SMB/CIFS Passwort Datenbanken

Einige Module um das System zusätzlich zu sichern:
`pam_cracklib` - Modul zum überprüfen und verhindern von unsicheren Passwörtern
`pam_passwdqc` - Ein weiteres Modul zum Überprüfen und Verhindern von unsicheren Passwörtern
`pam_securetty` - Erlaubt rootlogin nur von sicheren Devices aus die im File `/etc/securetty` deffiniert sind.
`pam_tally` - Zum Sperren von Benutzern bei zu viele Fehlversuchen.
`pam_access` - Über dieses Modul kann der Zugriff über den verwendeten login Name, Rechner Name, Domain Name, IP-Adresse oder das Terminal eingeschränkt werden.
`pam_time` - Zur Deffinition von Zeitspannen an denen für gewisse user kein login erlaubt ist.
`pam_warn` - Protokolliert: service, terminal, user, remote user and remote host.
`pam_rsa` - Ermöglicht RSA Authentication mittels Key auf einem Wechseldatenträger.
`pam_mount` - Kann verwendet werden um verschlüsselte Fielsysteme automatisch zu mounten. Dabei wird das verwendete Loginpasswort zum Entschlüsseln verwendet.
`pam_otpw` - Zur Verwendung von one time passwords.
`pam_abl` - Verwalten einer Liste von zu blockenden Rechnern.

3.4 Anzahl der Versuch berschränken - Ein Beispiel aus der Praxis

```
auth  required    pam_securetty.so
auth  required    pam_tally.so      per_user
auth  required    pam_nologin.so
auth  required    pam_env.so
auth  required    pam_mail.so
auth  sufficient  pam_krb5.so
auth  required    pam_unix.so       md5 use_first_pass
```

Zuerst wird überprüft von welchen Device aus zugegriffen wird z.B. `tty1`. root login wird nur erlaubt wenn das Device in `/etc/securetty` deffiniert ist. Da das Modul `required`

ist, wird mit dem darauffolgenden Modul weitergemacht. Das Modul `pam_tally.so` dient dazu den User nach zu vielen Fehlversuchen zu sperren. Als nächstes wird kontrolliert ob der Rechner durch das File `/etc/nologin` gesperrt ist, das Environment angepasst und die mails überprüft. Danach wird die Authentifikation mittels Kerberos 5 versucht. Bei einem Erfolg wird das Programm beendet, ansonsten wird mit dem Modul `pam_unix` fortgefahren.

```
account  sufficient  pam_krb5.so
account  sufficient  pam_unix.so
```

Es genügt wenn eines der beiden Module erfolgreich ist.

```
password  required    pam_pwcheck.so
password  sufficient  pam_krb5.so     use_first_pass use_authtok
password  required    pam_unix.so     use_first_pass use_authtok
```

Er wird das Modul `pam_pwcheck.so` verwendet um das neue Passwort zu überprüfen. Nach Erfolg wird zuerst versucht den Kerberosaccount upzudaten, wenn das misslingt wird der Unixaccount versucht.

```
session  required  pam_limits.so
session  required  pam_unix.so
session  optional  pam_krb5.so
```

Durch das Modul `pam_limits.so` werden die Resoursen für den User gegebenfalls eingeschränkt, dannach wird noch `pam_unix.so` und `pam_krb5.so` abgearbeitet.

4 Bitte draußen bleiben.

4.1 Endlich im Netzwerk

Mit der Sicherung der Zugriffsrechte und der generellen Härtung des Systems gegen lokale Angreifer ist schon mal ein erster großer Schritt geschafft. Wenn das System aber an das Internet angeschlossen ist, oder sogar Gateway für ein ganzes Subnetz bildet, dann reichen lokale Rechte nicht aus.

In der Vergangenheit gab es schon öfter Programme die über Netz-Angriffe dem Angreifer Tür und Tor öffneten und es ihm erlaubten die Kontrolle über den Benutzer zu bekommen mit dem das Programm ausgeführt wurde. Manchmal sogar über den root Benutzer.

Der erste Schritt ein System gegen Netzangriffe zu sichern sollte es sein, Programme die nicht über das Netz gebraucht werden an lokale Interfaces zu binden. Die meisten Programme lassen sich über den Bind Parameter an ein Interface oder eine IP-Adresse binden, also zum Beispiel an das Loopback Interface oder die 127.0.0.1 Adresse. Damit ist ein Programm über die externen Netzwerkinterfaces wie Ethernet oder Wlan nicht erreichbar.

4.2 Den Rechner verstecken

Da einige Services wie z.B.: http aber von anderen Computern aus erreichbar sein sollen/müssen, wird versucht möglichst wenige Angriffspunkte zu bieten und das System bei Problemen bzw. Angriffen leicht von anderen Subnetzen bzw. dem Internet isolieren zu können.

Mit Hilfe von Firewalls werden alle Verbindungsanfragen bewertet und dementsprechende Reaktionen durchgeführt. Computer im Subnetz werden vor Angriffen von Außen geschützt solange die Firewall standhällt
Es werden 2 grundlegende Arten von Firewalls unterschieden: Netzwerkschicht-Firewalls oder Paketfilter und Application-Proxy-Firewalls oder Anwendungsschicht-Gateways

Application-Proxy-Firewalls oder Anwendungsschicht-Gateways: Anwendungsschicht-Firewalls wie beispielsweise FireWallToolKit[1] können Daten der verwendeten Programme filtern und auf deren Basis die Filterentscheidungen treffen. Nachteil: sehr kompliziert zu implementiere da für den Datentransfer jedes Programmes eigene Regeln erstellt werden müssen.

Netzwerkschicht-Firewalls oder Paketfilter: Dabei handelt es sich im Wesentlichen um einen Router mit Paketfilterfunktionen.

Paketfilter netfilter/iptables: Seit der Version 2.4 sind netfilter und iptables (Dienstprogramm zur Konfiguration von netfilter) ein Bestandteil des Linuxkernels. Die fol-

[1] `http://www.fwtk.org/`

genden Auszüge eines Scriptes zeigt wie mit Hilfe von iptables Firewallregeln aufgestellt werden können.

Bei der Erstellung von Firewallregeln gibt es zwei konträre Ansätze:

Gutgläubigkeit: Ich erlaube alles und verbiete explizit Pakete

Paranoia: Ich verbiete alles und erlaube gewisse Pakete

In diesem Fall: Alles was über die Firewall hinein, hinaus oder weitergeleitet werden soll wird verboten!

```
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
```

Das zum Austausch von Informations- und Fehlermeldungen dienende Internet Control Message Protocol, kurz icmp, wird nun mit einer Ausnahme erlaubt. Dabei handelt es sich um die Beantwortung der Frage *Ist da jemand?*, dem echo request (ping). Er ist vom icmp-type 8. Durch blockieren des Ping ist der Rechner im Netz nicht auf den ersten Blick erkennbar, wahllose Attacken auf irgendeine IP können so vermieden werden.
`-A Input`: hängt einen Eintrag an die Regelkette für den Input an.
`-i $INTERFACE`: gibt die Netzwerkschnittstelle an
`-p`: gibt den Protokollnamen an
`-j DROP`: gibt an was mit dem Paket geschehen soll, in diesem Fall: verwerfen
Da die Regeln der Reihe nach abgearbeitet werden, wird zuerst ping verboten, danach icmp erlaubt.

```
$IPTABLES -A INPUT -i $INTERFACE -p icmp --icmp-type 8 -j DROP
$IPTABLES -A FORWARD -i \$INTERFACE -p icmp --icmp-type 8 -j DROP
$IPTABLES -A INPUT -j ACCEPT -p icmp
$IPTABLES -A OUTPUT -j ACCEPT -p icmp
$IPTABLES -A FORWARD -j ACCEPT -p icmp
```

Für den in `$ALLOW_SSH` definierten Rechner wird nun eine SSH Verbindung gestattet. Da nur mehr Pakete von einer gewissen IP angenommen werden, wird ein Angriff schwieriger. Unerwünschte bzw. unsichere Protokolle können so geblockt werden.

```
$IPTABLES -A INPUT -p tcp -i $INTERFACE -dport 22 -sport 1024:65535 -s $ALLOW_SSH
-m state -state NEW -j ACCEPT
```

4.3 Portscans

Was ist ein Portscanner?
Gezielte Anfragen auf Portnummern um herauszufinden ob eine Antwort erfolgt und d.h. ein Angriff möglich wäre.

Service Name	Protokoll	Port	Service Name	Protokoll	Port
ftp-data	TCP	20	ftp-command	TCO	21
ssh	TCP	22	telnet	TCP	23
smtp	TCP	25	dns	UDP	53
http	TCP	80	pop3	TCP	110
ntp	UDP	123	imap3	TCP	220
http3	TCP	443	imap-ssl	TCP	585

Tabelle 4: Häufig verwendete Netzwerk Services

Portscanns zu erkennen ist sehr schwer, da schon wenige, über einen großen Zeitraum verteilte, gezielte Anfragen einem möglichen Angreifer erlauben offene Ports zu finden.

Weniger trickreiche können allerdings mit Tools wie *portsentry* erkannt werden. Falls z.B.: von einer bestimmten Adresse mehrere Anfragen auf geschlossene Ports gestellt werden, so kann über einen Schwellwert diese IP für eine gewisse Zeit blockiert oder gegebenfalls gesperrt werden.

Scanner wie nmap oder SAINT zählen dennoch zu den unverzichtbaren Sicherheitstools, da sie in der Lage sind innerhalb kurzer Zeit sehr viele Rechner auf mögliche Angriffspunkte zu untersuchen und um Sicherheitsmechanismen, wie Firewallregeln zu testen.

4.4 Spoofing

Was ist Spoofing?
Als Spoofing werden Methoden bezeichnet die Identifikations- und Authentifizierungsmethoden zu untergraben versuchen.

Ansätze dazu gibt es viele: *IP-Spoofing, ARP-Spoofing, DNS-Spoofing* um nur einige zu nennen....

Der Grundgedanke ist allerdings immer der Selbe: versuche dich als jemand auszugeben der du nicht bist um an Informationen zu gelangen.

Bei *IP-Spoofing* beispielsweise wird im TCP-Header die Quelladresse der Pakete verändert. Oft wird dabei einfach versucht vorzugaukeln dass die Pakete aus dem eingenen Netz stammen würden. Durch blocken aller Pakete die bei der Firewall mit privaten IP-Adressen oder mit einer Loopback Adresse vom Internet eintreffen kann die Gefahr bereits verringert werden. Diese Adressen sind im Internet nicht zulässig und daher gefälscht.

Der Kernel selbst bietet mit `rp_filter` (siehe 1) ebenfalls einen Schutz gegen Spoofing.

Listing 1: Aktivierung des `rp_filters`

```
spoofing protection
for i in /proc/sys/net/ipv4/conf/*/rp_filter; do
        echo 1 > $i
done
```

Deaktivieren von sogenannten source routed packets (siehe 2), die zum Umgehen der Routing Tabellen verwendet werden können stellt eine weitere Möglichkeit dar.

Listing 2: Verwerfe alle source routed packets

```
disable source routed packets
for i in /proc/sys/net/ipv4/conf/*/accept_source_route; do
        echo 0 > $i
done
```

Bei *ARP-Spoofing* möchte der Angreifer seine eigene Hardwareadresse behalten, aber die IP eines anderen Hosts annehmen.

Eine mögliche, allerdings sehr aufwendige Methode zum Schutz ist die Verwendung von statischen ARP Einträgen, wie das Speichern von IP und Mac-Adressen in den ARP-Cache.

DNS-Spoofing verwendet den DNS Server um falsche Zuordnungen von Hostnamen und IP-Adressen zu erzeugen.

Werden alle Namen direkt von einem Nameserver aufgelöst, er hat sie also nicht in einem temporären Cache, sondern dauerhaft abgespeichert, kann diese Gefahr verringert werden.

Durch gezielte Paketfilterung kann die Netzwerksicherheit um einiges erhöht werden. So kann beispielsweise das blockieren von ausgehenden und eingehenden Redirect anfragen verhindern dass ein Angreifer Pakete auf seinen Host umlenken um an Informationen zu gelangen.
ICMP: Typ 5 (Redirect): Alternative Routerinformationen können an den Host gesendet werden.

Den besten Schutz gegen Spoofing-Attacken schaffen allerdings sichere Authentifizierungs- sowie Verschlüsselungmethoden sowie das rasche Erkennen eines möglichen Angriffes.

Auch dafür gibt es für Linux tools[2], die verschiedenste Spoofing-Attacken erkennen können.

4.5 DOS Attacken

Was sind DOS Attacken?

Denial-of-Service-Attacken sind eine der beliebtesten und vielfältigsten Angriffsmöglichkeiten. Dabei wird das Ziel verfolg Hosts durch Bandbreitensättigung, Ressourcensättigung oder Herbeiführung von System- und Anwendungsabstürzen unzugänglich und somit arbeitsunfähig zu machen.

Dabei werden häufig Unstimmigkeiten, Fehler oder sonstige Schwachstellen in Protokollen ausgenutzt. Diese Attacken können sich gegen die Netzwerkhardware wie einen Router oder gegen Anwendungsprogramme wie z.B.: Mailserver richten.

Einige bekannte Angriffe: *Syn Flooding, Ping Flooding, Mailbombing* ... nur um einige zu nennen.

[2] `http://www.securityfocus.com/tools/142`

Verteidigungsmethoden

Es gibt keine allgemein gülltige Verteidigungsmethode, doch kann die Abwehrkraft des Systems gestärkt werden.
Beispielsweise durch

- keine Broadcast-Adressen zulassen
- filtern von ICMP- und UDP-Datenverkehr
  - ICMP: Typ 11 (Time Exceeded) ausgehend: Nachricht wird an den Quellhost gesendet wenn die time-tp-life für das Paket abgelaufen ist. Dies kann eine nützliche Information für einen möglichen Angreifer sein.
  - ICMP: Typ 3 (Destination Unreachable) ausgehend: Diese Antwort wird gesendet wenn ein Host auf dem angeforderten Protokoll oder Port nicht erreichbar ist.
  - ICMP: Typ 4 (Source Quench) ausgehend: Aufforderung die Datenrate zu reduzieren da der Buffer nicht ausreicht um die Pakete weiterzuleiten.
- Einspielen von Sicherheits-Patches und Kernel-Updates

4.6 Weitere Angriffe

Es gibt noch eine Reihe von anderen Angriffsmöglichkeiten, wie Sniffer, die beispielsweise Passwörter ausspionieren versuchen, Vieren, Trojaner und jede Menge von nützlichen Tools die, wenn sie in die falschen Hände geraten eine Menge Schaden anrichten können.

Doch zum Glück sind diese nicht so weit verbreitet wie bei anderen Betriebssystemen.

4.7 Konfigurationstools

4.7.1 intrusion prevention and detection systems

Intrusion detection: Einbrüche in eine System erkennen

Wenn alle Sicherheitsmechanissmen fehlschlagen, der Angreifer in das System eingedrungen ist und möglicherweise Schaden angerichtet hat hillft ein intrusion detection system festzustellen was geändert oder manipuliert wurde. Eine Möglichkeit dafür sind die Checksummen von Konfigurationsfiles nach der Installation zu speichern und regelmässig zu überprüfen.

Intrusion prevention: Einbrüche verhindern

Durch Auswerten von Logfiles verschiedener Prozesse können Angriffe und Einbrüche erkannt werden. Auch die Analyse des Netzwerkverkehrs kann Auskunft über Angriffe liefern. Dazu wird der Netzwerkverkehr auf Muster gefiltert. Einfache Angriffe können somit schnell erkannt und Gegenmaßnahmen eingeleitet werden. Zum Beispiel kann die IP-Adresse des

Angreifers geblockt oder Personen benachrichtigt werden.

Intrusion prevention and detection systems können das System leider nicht zu 100% schützen. Jedoch werden die meisten Angriffe mit bekannten Techniken ausgeführt die erkannt oder sogar abgewehrt werden können.

Einige intrusion prevention and detection Systeme

Snort [?]: Ist eines der am häufigst verwendetsten Tools. Es hat unter anderem eine Funktion die es erlaubt den gesammen Traffic bei einem erkannten Angriff mitzschneiden um ihn später analysieren zu können. Snort verwendet Protokoll Analysen und filtert den Inhalt nach Muster. Es existieren einige User-Interfaces für Snort zum dursuchen und analysieren der gesammelten Daten wie zum Beispiel das Webinterface *BASE* [?].

Bro Intrusion Detection System [?] : Filtert und analysiert Netzwerktraffic und kann Einbruchversuche anhand von Trafficcharakteristiken und dem Inhalt dedektieren. Bei einem erkannten Einbruchversuch können Scripte ausgeführt werden die Gegenmaßnahmen, wie zum Beispiel einen Prozess beenden oder die Firewall anweisse die IP-Adressen des Angreifers zu blocken, eingeleitet werden.

Prelude Hybrid IDS [?]: Wertet Logfiles und Informationen von Sensoren wie *Snort*, *honeyd* oder *Nessus* aus. Vorteil ist, dass Daten von verschiedenen Systemen ausgewertet werden und so eventuell ein Angriff auch noch erkannt wird, auch wenn er von den anderen Systeme nicht erkannt werden konnte.

4.7.2 firewall configuration tools

Firewall Builder [?]: Firewall Builder ist ein configurations und management Tool das einen Objektorientierten Ansatz ermöglicht. Es beinhaltet Policy-Kompiler für verschiedene Firewalls, unter anderen auch iptables. Das Userinterface ermöglicht es die Firewall mittels Drag&Drop zu konfigurieren.

KMyFirewall [?]: Ist eine auf QT basierende GUI um leicht eine Personal Firewall auf Basis von iptables zu konfigurieren. Es bietet 2 Interfaces, eines für experten und eines für alle anderen. Die automatisch erzeugten Skipte können direkt in das Init-System integriert werden.

FireHOL [?]: Ist eigentlich eine Skriptsprache um Firewall Regeln automatisch zu erstellen. Da die Konfigurations-Files sind BASH Skripte sind können einfach andere Skripte und Programme eingebaut werden. Die Verwendung von Variablen, Pipes und Schleifen ist auch möglich. FireHOL ist auch für sehr komplizierte Konfigurationen geeignet.

5 Entwurzelung, nieder mit dem Herrscher

Wie gezeigt, bietet Linux sehr gute Mechanismen ein System abzusichern. Wenn ein System besondere Sicherheits- oder Datenschutzregeln einhalten muß, dann reichen die grundlegenden Linuxmechanismen nicht mehr aus.

Größte Schwachstelle eines voll abgesicherten Systems ist meistens der Root-Benutzer. Er hat in einem Standard Linux System alle Rechte, kann Programme installieren, deinstallieren, Rechte verändern und sich Zugang zu allen Daten auf dem System beschaffen. Das ist nicht immer so gewünscht. Zum Beispiel kann es in einem öffentlichen Büro wünschenswert sein dass ein Angestellter der IT-Abteilung keinen Zugriff auf geheime Dokumente hat.

5.1 SELinux

Nicht umsonst wurde der hier nun vorgestellte Ansatz von der National Security Agency der USA entwickelt. SE-Linux, Security Enhanced Linux wurde ursprünglich von der NSA entwickelt wird seit der Integration in den Offiziellen Kernel auch von verschiedenen Firmen wie RedHat, Network Associates, Secure Computing Corporation, und Tresys mitentwickelt. Es besteht aus einer Kernelkomponente und den ladbaren Policy Regeln. Über die Policies kann der Zugriff auf die Ressourcen eines Systems geregelt werden. Dabei unterstützt SELinux eine Vielzahl von verschiedenen Sicherheitstechniken:

5.1.1 Mandatory Access Control

Mandatory Access Control(MAC) ist der Überbegriff für das von SELinux bereitgestellten Sicherheitskonzepte. Das Standard-Linux Konzept regelt den Zugriff über die Identität des Benutzers(oder Prozess) und der Ressource(Datei) auf die Zugegriffen werden soll. MAC erweitert dieses Konzept, der Zugriff wird auch noch durch zusätzliche Regeln und Eigenschaften festgelegt, die in der Policy spezifiziert werden.

5.1.2 MLS/MCS

Dabei werden beim MAC-Model zwei Varianten unterschieden, Multi-Level Security und Multi-Lateral Security, wobei SELinux die Multi-Level Security nach Bell-LaPadula implementiert. Dabei wird der Zugriff in mehrere Schichten eingeteilt, zum Beispiel Öffentlich oder Privat. Nach Bell-LaPadula soll es in einer niederen Schicht wie Öffentlich, nicht möglich sein Daten aus einer höheren Schicht wie Privat zu lesen. Standardmäßig heißen diese Level in SELinux SystemHigh und SystemLow.

Auf das Multi Level Security baut die Multi Category Security auf. Aber alle Objekte bekommen bei MCS die selbe Sicherheitssensitivität unterscheiden sich aber in ihrer Kategorie. Kategorien können im Unterschied zur Sicherheitssensitivität vom Benutzer vergeben werden. Seit Kernel 2.6.12 unterstützt unterstützt SE-Linux auch MCS. Die Kategorien heißen c0-c255. Sie sind standardmäßig in der Referenzpolicy aber nicht aktiviert.

5.1.3 Role Based Access Control

Beim Role Based Access Control(RBAC), der rollenbasierten Zugangskontrolle werden den Benutzern eine oder mehrere Rollen zugewiesen. Die Zugriffsrechte sind diesen Rollen zugewiesen. Dies kann die Administration erleichtern. RBAC kann auch zusammen mit MAC verwendet werden, was aber die Administration erschwert, da sich die MAC und RBAC Rechte überschneiden bzw widersprechen können. RBAC kann aber auch als Abstraktionsebene des MAC Modells dienen und dadurch die Abstraktion vereinfachen. Standardmäßig gibt es in 3 User:
root für den root-User
system_u für Systemprozesse
user_t für User die keine SELinux Identität haben

Die Standard Rollen sind:
staff_r von Benutzern verwendet die zu sysadm_r wechseln können *sysadm_r* für den System Administrator *system_r* für Systemprozesse *user_r* für normale Benutzer die nicht zu sysadm_r werden können

5.1.4 Type Enforcement

Type Enforcement bildet die Grundlage für die vorgestellen Sicherheitsmodelle. Dabei werden jedem Subject und jeder Ressource ein Typ-Attribut zugeordnet. Bei den Typen spricht man bei Benutzern und Prozessen von Domain oder Sandbox und bei Ressourcen wie Files von Attributen. Tecvhnisch gibt es in SELinux zwischen Domain und Attribut keinen Unterschied. Durch die Type Enforcement Matrix kann man so feststellen welches Subjekt Zugriff auf welches Objekt hat.

Es gibt 7 Arten von TE Deklarationen.

attribute_def: Attributdeklaration Attribute werden Ressourcen vergeben um diese in Gruppen zusammenzufassen. Die Definition geschiet einfach durch
attrib admin
Damit wird admin als Attribut festgelegt.

type_def: Typdeklaration Ein Beispiel für eine Type Deklaration sieht folgendermaßen aus:
type ping_t, domain, privlog;
Der Befehl setzt dem Typ ping_t die Attribute Domain und Privlog, mit denen der Typ als Domain angelegt wird der mit dem Syslog-Prozess kommuniziert.

Die Definition für den Syslog-Prozess sieht folgendermaßen aus:
allow privlog devlog_t:sock_file { ioctl read getattr lock write append };
allow privlog syslogd_t:unix_dgram_socket sendto;
allow privlog syslogd_t:unix_stream_socket connectto;
allow privlog devlog_t:lnk_file read;
Damit wird Syslog das Recht gegeben um auf Socketfiles, die das devlog_t Attribut

haben, zuzugreifen und auf Sockets, die das syslogd_t Attribut besitzen, zu lesen und zu schreiben. Der letzte Befehl gibt Prozessen mit dem Privlog Attribut Rechte Dateien und Symbolische Links mit dem devlog_t Attribut zu lesen.

typealias_def: Typalias deklaration Typalias-Deklarationen erlauben es Aliase für Typ-Deklarationen zu erstellen. Beispiel:
typealias cupsd_etc_t alias etc_cupsd_t;
Definiert etc_cupsd_t als Alias für cupsd_etc_t. Die erleichtert die Administration und die Arbeit mit fremden Policy Files.

te_avtab_def: TE Zugriffsvektortabellen Deklaration Über die Zugriffstabelle können Rechte vergeben werden. Rechte können sein: *allow*: Zugriff wird erlaubt
auditallow: Zugriff wird erlaubt und geloggt
auditdeny: Zugriff wird verweigert und geloggt
dontaudit: Verhindert dass Zugriffe geloggt werden
Beispiel:
allow ping_t ping_exec_t:file { read getattr lock execute ioctl };
Erlaubt Prozessen in der pint_t Sandbox alle Rechte an Dateien mit ping_exec_t Attributen.

transition_def: Übergangdeklaration Transition Deklarationen regeln den Übergang von eine Sandbox in eine andere. Das Beispiel:
type_transition sysadm_t ping_exec_t:process ping_t;
zeigt dass ein Programm in der sysadm_t Sandbox, das die ping_exec Attribute hat, beim Ausführen versucht in die ping_t Sandbox zu wechseln.

bool_def: Boolean-deklaration Die Boolean Deklaration kann dazu verwendet werden Zugriff von Unpreviligiertern Benutzern zu erlauben oder sperren. Für Ping wär ein Beispiel:
bool user_ping false;

cond_stmt_def: Bedingungsstatementdeklaration Conditional Statements erlauben if-Abfragen in der Policy, z.B.
if(user_ping) dann ...

5.1.5 CAP - Linux Capabilities

Eine einfachere Möglichkeit die Sicherheit eines Systems zu verbessern wird auch durch die Capabilities geboten. Capabilities gibt es im Kernel schon seit 2.2.11. Allerdings können diese nur mit modernen Patches wie SE-Linux, RSBAC, AppArmor usw. genutzt werden.

Gesetzt und gelesen können System CAP's über das /proc Dateisystem, über */proc/sys/kernel/cap-bound*. Einige Distributionen bieten Tools die die Verwaltung vereinfachen, z.B. lcap.

Aber es ist Vorsicht geboten. Capabilities die einmal entfernt werden können nicht so einfach wieder gesetzt werden, auch nicht vom root-User. Meist hilft da nur ein Reboot. Allerdings ist der Einsatz durchaus effektiv sein. Ein fertig konfiguriertes und gestartetes System kann durch entfernen von *CAP_SYS_MODULE* daran gehindert werden, weitere Module nachzuladen. Eine Netzwerkfirewall die mit statischen Verbindungen betrieben wird, kann durch *CAP_NET_ADMIN* Veränderungen an der Netzkonfiguration verhindern.

Aber CAP's gelten nicht nur für Prozesse, es gibt auch Capabilities für das Filesystem. So können z.B. bestimmten Befehle, die bisher SetUID-Root Rechte benötigen und damit leicht für Angriffe mißbraucht werden können, besser abgesichert werden. Ein einfaches Beispiel ist der ping befehl. Um von jedem Benutzer ausgeführt werden zu können und Zugriff auf die Netzwerk-Sockets zu erhalten hat ping SUID Rechte. Durch die Filesystem CAP's braucht ping keine SUID Rechte mehr sondern nur die *CAP_NET_RAW* Capabilities.

Außerdem kann man auch bestimmte Prozesse die bisher als Root gestartet werden mußten, z.B. Samba, Apache, DHCPD, Bind, ... als normaler Benutzer ausführen. Weitere informationen bietet [FRIED]

5.1.6 Alternativen

Alternativen zu SELinux gibt es einige in aber die müssen erst nachgepached werden. Dazu gehören: AppArmor, von SUSE/Novell. Es gilt als einfacher konfigurierbar als SELinux, aber weit nicht so mächtig. Bietet dafür aber gute Perfomance. Größter Kritikpunkt ist dass Dateien über Ihren Dateinamen übernommen werden. GRSecurity bietet RBAC für den Linux Kernel, außerdem noch Chroot Security, Randomisierung des User und Kernelstacks, Bibliotheken und Heap und einiges mehr RSBAC entstand aus einer Diplomarbeit von Amon Ott. Es stellt ein Framework für verschiedene Sicherheitsmodelle bereit und integriert auch Dazuko als Antivirus in den Kernel. Es ist noch mächtiger als SELinux, setzt aber nicht auf die Linux Security Modules im Kernel auf und hat damit nur geringe Chancen in den offiziellen Kernel aufgenommen zu werden. Vom Unabhängige Landeszentrum für Datenschutz (ULD) in Deutschland hat TightgatePro eine Appliance die RSBAC integriert ein Datenschutzzertifikat bekommen.

6 Fazit

Alle Möglichkeiten und Ansätze, die Linux bietet um ein System abzusichern, zu erklären würden ganze Bücher füllen, deshalb wurden hier Themen auch nur angerissen um einen kleinen Einblick in die Grundkonzepte zu gewähren. Linux bietet sehr gute Möglichkeiten ein System abzusichern, allerdings muss man sich mit dem Betriebssystem auseinandersetzen.

Sprich man muss hinter die Kulissen der grafischen Benutzeroberflächen blicken, um sich in die tiefen, unendlichen Weiten von Kernel, Modulen, und Skripten ein sicheres System zu bauen das, zumindest meistens, genau das macht was man will.

Ob es dabei aber bei der Mensch-Maschinenkommunikation Missverständnisse gibt ist eine andere Geschichte, denn 99% aller Fehler sitzen 60cm vom Monitor entfernt...