

**Root Passwort unter Linux zurücksetzen**

**Inhalt**

**1. Einleitung**
**2. Der Bootvorgang**
**3. Schutzmassnahmen**
**4. Wir setzen das Root Passwort zurück**
**5. Mini Glossar**

## 1. Einleitung

Nach einer langen Pause gibt es wiedereinmal ein Tutorial von mir.
Heute möchte ich euch einen Weg zeigen, wie ihr das root Passwort auf einem lokalen Linux System zurücksetzen könnt. Dieses Tutorial soll Hilfreich sein, dass man trotz Verlust des root Passwortes wieder root Rechte erlangen kann.
Ich möchte dabei gleich zu Anfang erwähnen, dass es sich bei der hier besprochenen Methode nicht um eine Sicherheitslücke handelt.
Wir werden durch den Bootmanager die **Bash** aufrufen und von dort aus versuchen ein neues Passwort zu setzen.
Um das ganze jedoch besser verstehen zu können, halte ich es für sinnvoll zuerst auf den Bootvorgang eines Linux Systems einzugehen.
Weiterhin möchte ich einige Schutzmassnahmen besprechen, bevor wir zum eigentlichen praktischen Teil wechseln werden.
Ich möchte an dieser Stelle auch in Erinnerung rufen, dass es keine 100%tige Sicherheit gibt. Grundsätzlich ist ein System von dem Moment an unsicher, indem man die Box physisch berühren kann ;-)




Ich bitte euch auch, dass ihr das hier erwähnte nicht missbraucht um einem System zu Schaden und erwähne ausdrücklich, dass jeder für sein Handeln selbst verantwortlich ist.
Nun wünsche ich euch viel Spass beim lesen und hoffe, dass ihr etwas lernen könnt.

## 2. Der Bootvorgang

Was passiert eigentlich alles beim Starten?
**Dos** kennt nur die config.sys und die autoexec.bat, aber bei Linux läuft das jedoch alles ein wenig vielseitiger ab. Nach dem Einschalten des Rechners und dem Durchlaufen des **POST** (Power On Self Test) wird normalerweise von der ersten IDE-Festplatte der Bootsektor ausgeführt. Unter Linux wird hierfür meistens der **LILO** (Linux Loader) oder **GRUB** verwendet. Der Bootmanager kann verschiedene Betriebssysteme auch von unterschiedlichen Festplatten starten oder einem Linux-Kernel gewünschte Parameter übergeben. Der **Kernel** ist prinzipiell für die Erkennung und Einbindung der gesamten Hardware zuständig und führt bei jedem Booten eine Hardwareerkennung durch. Ein für Pentium optimierter Kernel läuft sowohl auf Pentium I, II, III und Athlon/Duron. Man muss also nach dem Aufrüsten eines Rechners (Mainboard, Prozessor, Speicher) das System nicht neu installieren.
Die letzte Aufgabe des Kernels ist es, die Partition, die ihm als root-Partition vorgegeben wurde zu **mounten** und den ersten Prozess zu starten, der normalerweise **INIT** heisst (/**sbin**/**init**). Jetzt ist der Kernel prinzipiell mit Laden fertig und die gesamte Hardware (die er eingebunden hat) steht zur Verfügung. Die Meldungen, die Kernel und Programm trennen, lauten:

```
VFS: mounted root (ext2fs) filesystem readonly. (noch kernel)
INIT: Version X.XX booting (das ist INIT)
```

Das Init-Programm liest seine Konfiguration aus der Textdatei "/**etc**/**inittab**". In dieser Datei wird der Standard-Runlevel festgelegt:

```
id:3:initdefault:
```

Als nächstes folgt die Angabe des **Shellscripts**, welches beim Starten des Rechners ausgeführt wird:

```
si::sysinit:/etc/init.d/rcS
```

Dieses Script sorgt unter anderem dafür, dass die Dateisysteme überprüft, die in der /**etc**/**fstab** stehenden Partitionen gemountet werden und die Systemzeit gestellt wird. Anschliessend stehen die Script-Aufrufe für die **Runlevel**:

```
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

Wurden also sowohl das **Sysinit-Script**, als auch das entsprechende Script für den **Standard-Runlevel** ausgeführt, werden meistens sechs virtuelle Konsolen gestartet:

```
1:23:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
```

Ein Runlevel legt fest, welche Dienste gestartet werden und welche nicht. Runlevel "1" bzw. "S" ist der Single-User-Modus ohne Netzwerk.
Runlevel "2" soll Multi-Userfunktion mit Netzwerkunterstützung bieten.
Runlevel "3" Netzwerkunterstützung mit Netzwerkdiensten (Apache, MySQL, ...) und Runlevel "5" mit grafischer Anmeldeoberfläche (xdm, kdm oder gdm).
Runlevel "0" bedeutet "Rechner herunterfahren", Runlevel "6" steht für "Rechner neustarten". und Runlevel "4" wird nicht genutzt.
Die Start- und Stop-Scripte können auch manuell aufgerufen werden, um während des Betriebs einen Dienst herunterzufahren bzw. zu starten.

```
etc/init.d/dienst restart
```

### 3. Schutzmassnahmen

Bisher haben wir einen Überblick erhalten, wie der Bootvorgang eines Linux Systems abläuft und in der Einleitung wurde angetönt, dass wir durch den Bootmanager die Bash aufrufen werden um von dort aus das Passwort zu ändern.

Wir haben also vor das übliche Bootprogramm und die Authentifizierung zu umgehen und statt der standardmässigen init .die bash zu laden.
Da dieser Eingriff vor dem Bootvorgang stattfindet, werden wir uns mal Grundlegend darüber Gedanken machen, was wir tun könnten, um unser System besser abzusichern.

### 3.1. Power On Passwort/Bios Passwort setzen

Aktivieren Sie in Ihrem Bios die Power On Passwort Funktion, damit das System vor dem Booten zusätzlich mit einem Passwort geschützt ist.
Weiterhin ist es wichtig, das Bios ebenfalls mit einem Passwort zu versehen, da die vorherige Massnahme sonst eher nutzlos wäre.
Es gibt einige Bios Hersteller, aber alle sollten diese Funktion unterstützen.
Die nachfolgende Grafik zeigt eine Abbildung des Phoenix Bios.

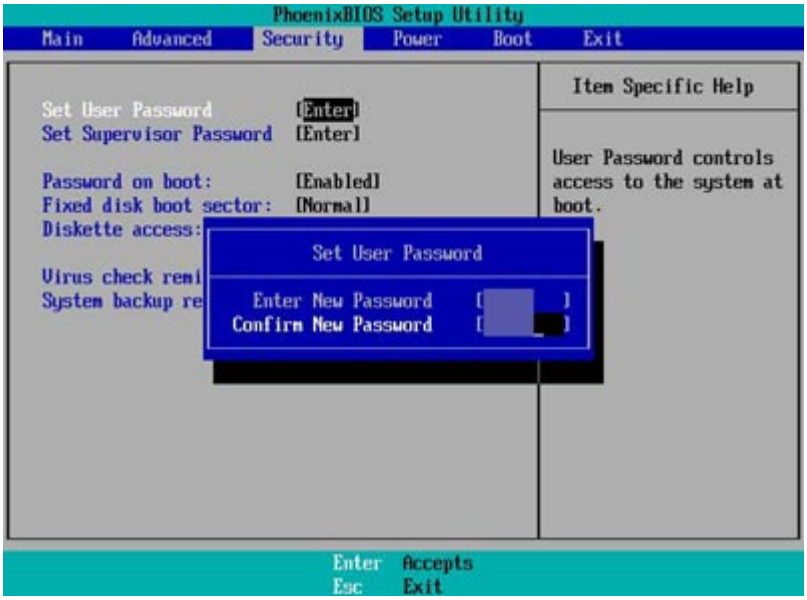



**Abbildung 1:Phoenix Bios**

### 3.2. Bootsequenz abändern

Um zu verhindern, dass von einem anderen Medium, als von der HD gebootet wird, sollte man als primäre Bootsequenz die HD wählen.
Diese Massnahme dient auch als Schutz gegen Infektion von Bootsektorviren.

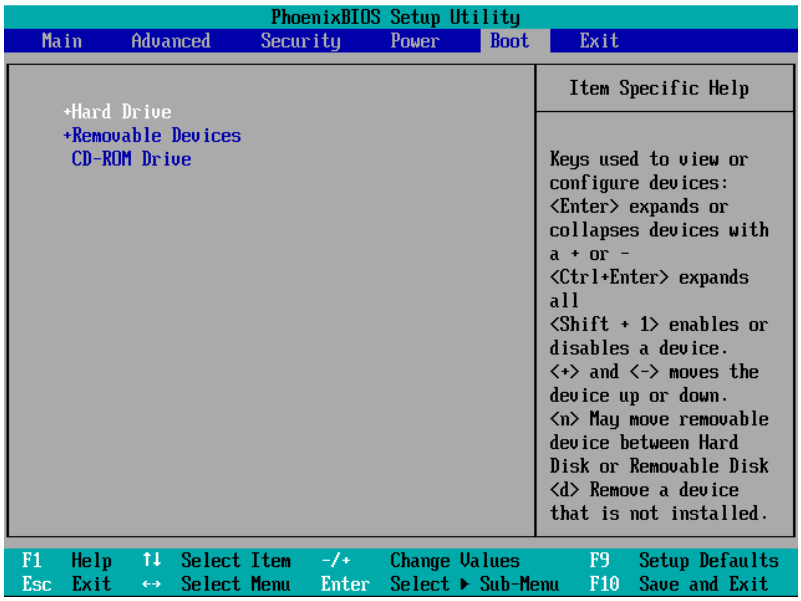



**Abbildung 2: HD als primäre Bootsequenz**

Man könnte denken Windows bietet eine bessere Sicherheit, da wir ja schliesslich keine Bash aufrufen können. Dabei handelt sich aber nur um eine Scheinsicherheit. Wie wir ja wissen ist es nicht unbekanntes, dass es komplette Bootbare Betriebssysteme ab CD gibt. Das klassischste darunter dürfte sicher Knoppix Linux sein.
Wir können so unabhängig von unserem installierten OS unser Knoppix starten und bekommen danach Zugriff auf alle Files.
Gerade aus diesem Grund dürfte es wichtig sein der HD erste Priorität zu geben.

Wer mehr zum Thema Bios und zu den einzelnen Einstellungen erfahren möchte, empfehle ich einen Besuch auf der Seite *http://www.bios-kompendium.de*

### 3.3. Bootparametereingabe im Bootmanager schützen

Natürlich lässt sich auch eine Massnahme treffen, dass die Bootparametereingabe, die wir uns nachher zu Nutze machen durch ein Passwort geschützt wird.
Dazu gehen wir in die Shell und lassen uns als erstes einen **Hashwer**t generieren.

***grub-md5-crypt***

```
linux:~ # grub-md5-crypt
Password:
Retype password:
$1$L7JL51$Ko1bFU2sA3OGCKwH2N/W8/
linux:~ # _
```

Danach werden wir nach einem Passwort gefragt und bekommen den Hashwert geliefert.

**Abbildung 3: Hashwert**

Diesen Hashwert müssen wir jetzt kopieren und später in der Grub Konfiguration einfügen. Ich habe dazu die Ausgabe mittels ***grub-md5-crypt >hash.txt*** in ein Textfile namens hash.txt gespeichert. Danach das neu erstellte File mit **vi** geöffnet und von dort den Hashwert kopiert.

Mittels **vi** öffnen wir nun ebenfalls ***/boot/grub/menu.lst***

```
color white/blue black/light-gray
default 1
timeout 8
gfxmenu (hd0,6)/boot/message

password --md5 $1$LYHL51$vEe9yjePc50/M78Dab7Ny.
```

**Abbildung 4: Hashwert in der Grubkonfiguration einfügen**

Hier müssen wir **noch *password --md5*** anfügen und danach unser **Hashwert** einfügen. Jetzt speichern wir unsere Konfiguration und können den Computer neu starten.

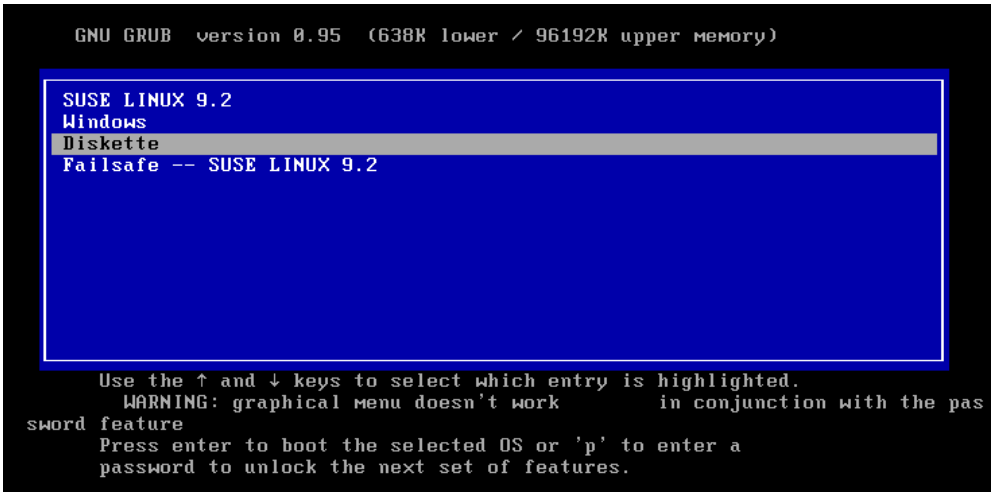



**Abbildung 5: Grub mit geschützter Bootparameterfunktion**

Wie wir nun erkennen ist die Bootparametereingabe durch ein Passwort geschützt und wir können nur noch das jeweilige OS auswählen.

### 3.4 Verschlüsseln der HD

Ich gebe zu diesen Teil bei der ersten Version etwas „stiefmüttlerlich“ behandelt zu haben. Trotzdem möchte ich bei dieser Schutzmassnahme nicht ins Detail eintauchen, sondern eher einen Denkanstoss liefern.

Daten können zusätzlich durch Verschlüsselung geschützt werden. Unter Verschlüsselung verstehe ich die Kodierung von Daten, so dass diese nur von denjenigen gelesen werden können, für die sie bestimmt sind.
Hier unterscheide ich zwischen mehreren Möglichkeiten und möchte jeden Punkt noch kurz Erläutern:

1. Verschlüsseln der ganzen Disk durch Hardware
2. Verschlüsseln der ganzen Disk durch Bios Erweiterung
3. Verschlüsseln von Partitionen
4. Verschlüsseln von einzelnen Files und Verzeichnistrees

Verschlüsseln der ganzen Disk durch Hardware dürfte relativ sicher, aber aufwendig sein. Da die wenigsten Mainboards ein Verschlüsselungsmodul eingebaut haben, müsste man sich zusätzliche Hardware anschaffen, was schnell einmal teuer werden dürfte. Ich habe im Internet mal nach einer diesbezüglichen Lösung Recherchiert und bin auf einen Crypto IDE-Controller der Firma Elkom[1] gestossen.

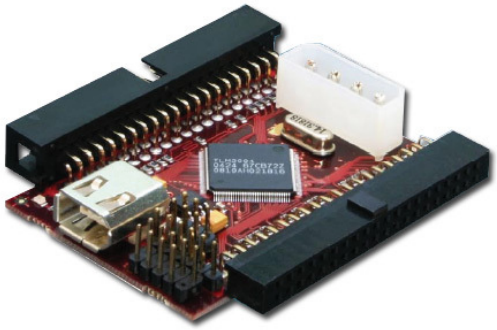


**Abbildung 6 Crypto-IDE Controller**

Es handelt sich dabei um ein Modul, welches zwischen die IDE HD gehängt wird und Daten in Echtzeit „on the fly“ ohne Einbussen der Systemperformance verschlüsselt. Die kompletten Daten inkl. MBR, Partitionstabelle und OS werden im 3-DES Algorithmus verschlüsselt und können nur mit einem sogenannten Token, der an den Firewireport gehängt wird wieder freigeschalten werden. Der geheime Teil des Schlüsselbundes, welches sich auf dem Token befindet gelangt niemals auf die HD oder in das RAM, so das dieser auch nicht ausspioniert werden kann.
Ich kann diesbezüglich aber keine Empfehlungen geben, da ich ein solches Hardware Modul bisher noch nie eingesetzt habe.

[1] http://www.elkom.com.tw

Bei den meisten Notebooks kann man die Festplatte mit einem Passwort versehen, um die Daten gegen Unbefugte zu schützen. Seit Microsoft diese Funktionen auch für die XBOX verwendet hat, werden diese auch von fast allen 3.5" Festplatten unterstützt. Die in den meisten PCs verwendeten BIOS Versionen bieten jedoch keine oder nur beschränkte Unterstützung der "Security Feature Set" Funktionen. Hierdurch entsteht eine Sicherheitslücke, die Schadprogramme nutzen können, um im Betrieb unbemerkt ein Festplattenpasswort zu setzen und somit beim nächsten Neustart den Festplattenzugriff und das Booten eines Betriebssystems verweigern. Ein sehr detaillierter Bericht dazu ist bei Heise[2] zu finden. Aus diesem Bericht geht auch hervor, dass dieser Schutz nicht gut genug ist und von Experten ausgehebelt werden könnte. Wer sich näher für eine solche Bios Erweiterung interessiert, dem Empfehle ich bei der Internetseite von Fitzenreiter[3] vorbeizuschauen.

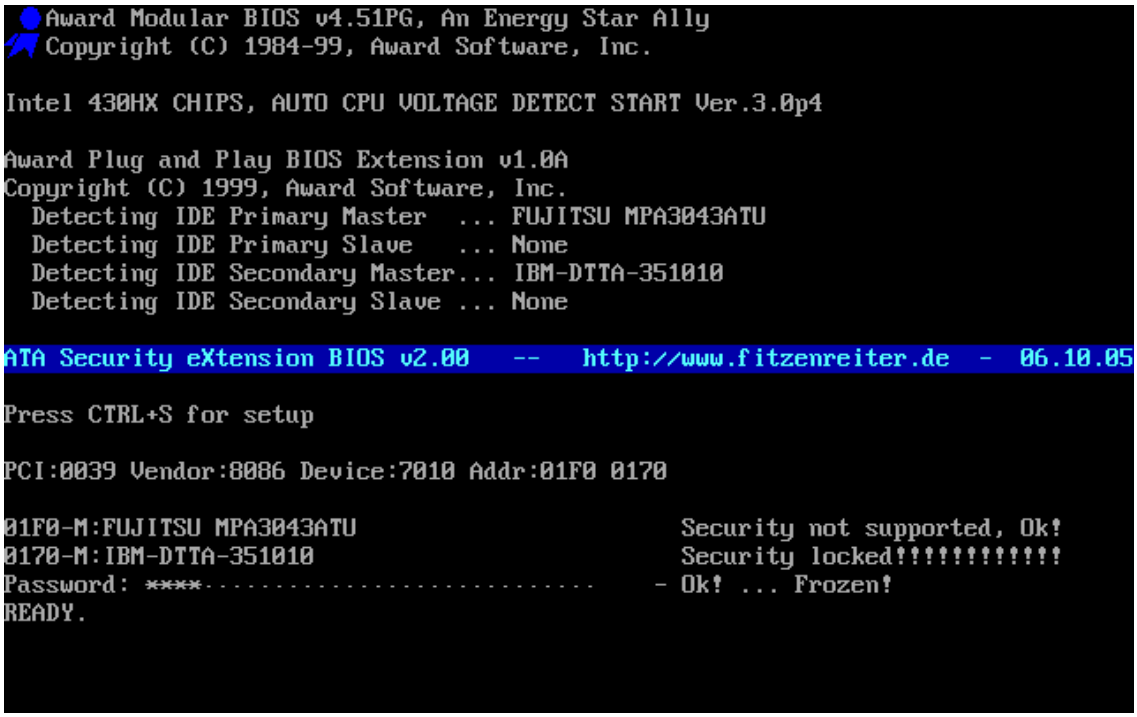



**Abbildung 7 Security Feature Set Bios Erweiterung**

Zur Verschlüsselung von Partitionen oder der ganzen HD gibt es mehrere Produkte. Eine Software, welche zur Verschlüsselung von Partitionen eingesetzt werden könnte ist z.B. Truecrypt[4]. Diese Software ist Open-Source und es gibt sie sowohl für Windows als auch für Linux. TrueCrypt verschlüsselt nicht einzelne Dateien sondern benutzt Container. Ein Container kann eine Festplattenpartition oder eine Datei von fast beliebiger Grösse sein. Wenn TrueCrypt einen Container öffnet, wird zuerst das Passwort abgefragt und dann erscheint ein neuer (virtueller) Datenträger im Windows-Explorer. Auf diesem Datenträger können wie gewohnt Dateien angelegt, umbenannt und gelöscht werden. Die Verschlüsselung geschieht ebenfalls „on the fly“, also praktisch ohne spürbare Verzögerung.
Auf eine gezielte Empfehlung möchte ich jedoch verzichten, da es viele Kryptographieprodukte gibt und jeder unterschiedliche Erwartungen an eine derartige Lösung stellt. Ich empfehle aber, dass man sich vor dem Nutzen einer Kryptographiesoftware unbedingt zuerst darüber Gedanken machen sollte, wie sie funktioniert.

[2] Bericht über Festplattensicherheit - http://www.heise.de/ct/05/08/172/
[3] Security Feature Set Bios Erweiterung - http://www.fitzenreiter.de/ata/ata.htm
[4] Truecrypt – http://www.truecrypt.org

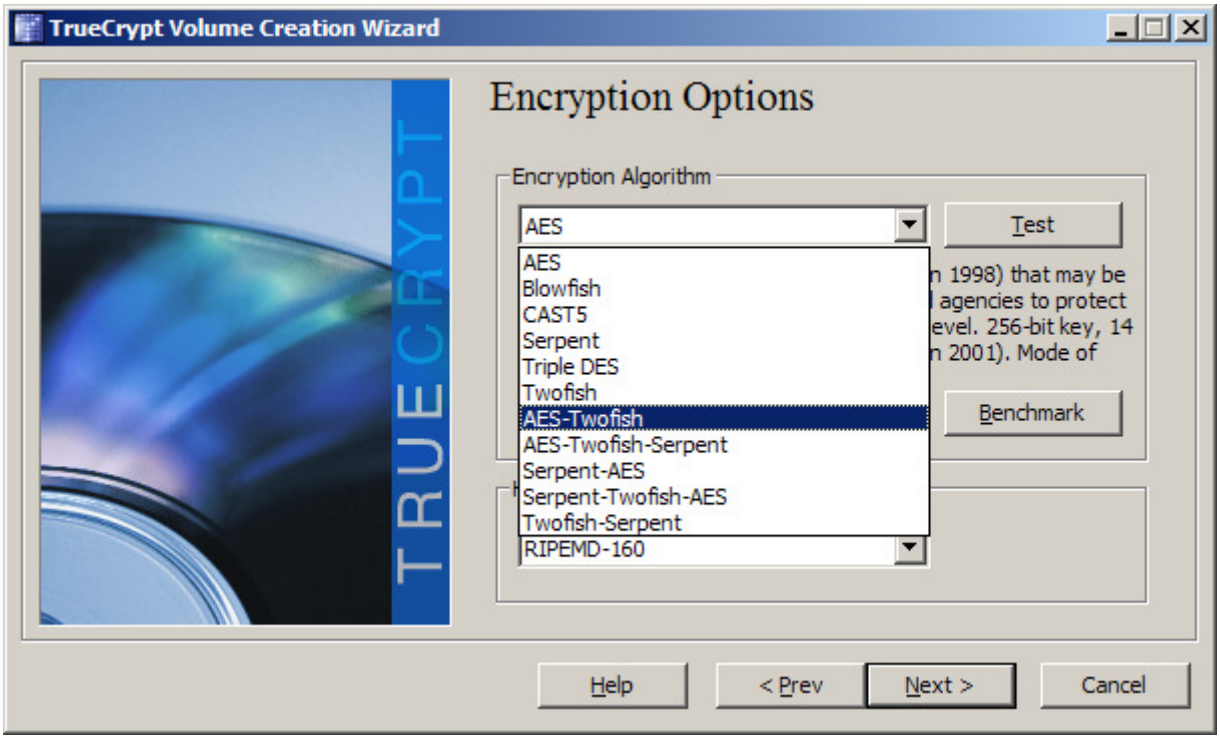



**Abbildung 8 TrueCrypt**

Bei der Verschlüsselung von einzelnen Files und Verzeichnistrees könnte man sich wieder einer Container Verschlüsselung bedienen oder den EFS Schutz unter Windows 2000 oder Windows XP Professional zu nutze machen. Vorraussetzung das dieser Schutz funktioniert ist lediglich, dass der Datenträger mit NTFS Formatiert wurde. Da ich nicht vor habe zu weit vom eigentlichen Thema abzutrifften möchte ich an dieser Stelle nicht erklären wie EFS funktioniert, sondern halte es für wichtig noch aufzuzeigen, dass EFS gewissen Beschränkungen unterliegt, über die man sich im klaren sein sollte.

- Mit EFS ist es nicht möglich, komprimierte Daten oder Systemdaten zu verschlüsseln. Diese Eigenschaft gewährleistet u.a. einen fehlerfreien Systemstart, da EFS zu diesem Zeitpunkt noch nicht aktiv ist.
- Hat ein unbefugter Nutzer Zugriff auf ein verschlüsseltes Verzeichnis, kann er zwar die darin enthaltenen Daten nicht entschlüsseln, aber sehr wohl sehen, um welche Daten es sich dabei handelt. Der Zugriff auf ein verschlüsseltes Verzeichnis bleibt beim Einsatz von EFS vollkommen unberührt.
- Zu beachten ist weiterhin, dass temporäre Dateien, die möglicherweise bei der Bearbeitung eines verschlüsselten Dokumentes entstehen, nur dann verschlüsselt werden, wenn diese in einem verschlüsselten Ordner liegen. Es bietet sich daher an, einen zentralen Ordner für solchen temporären Dateien zu erstellen und für diesen das Attribut zur Verschlüsselung zu aktivieren.

Sobald Daten mit EFS verschlüsselt werden, sollte unbedingt der private Schlüssel auf Diskette oder CDROM gesichert werden, da sonst bei einer Beschädigung bzw. Neuinstallation von Windows nicht mehr auf die verschlüsselten Daten zugegriffen werden kann. Selbst wenn man ursprünglich der rechtmässige Benutzer war!

Weiterhin sollte man sich im klaren sein, dass EFS geknackt wurde.
Der russische Krypto-Spezialist ElcomSoft[5] hat schon vor längerer Zeit eine Software entwickelt mit der es möglich ist EFS geschützte Dateien zu entschlüsseln.

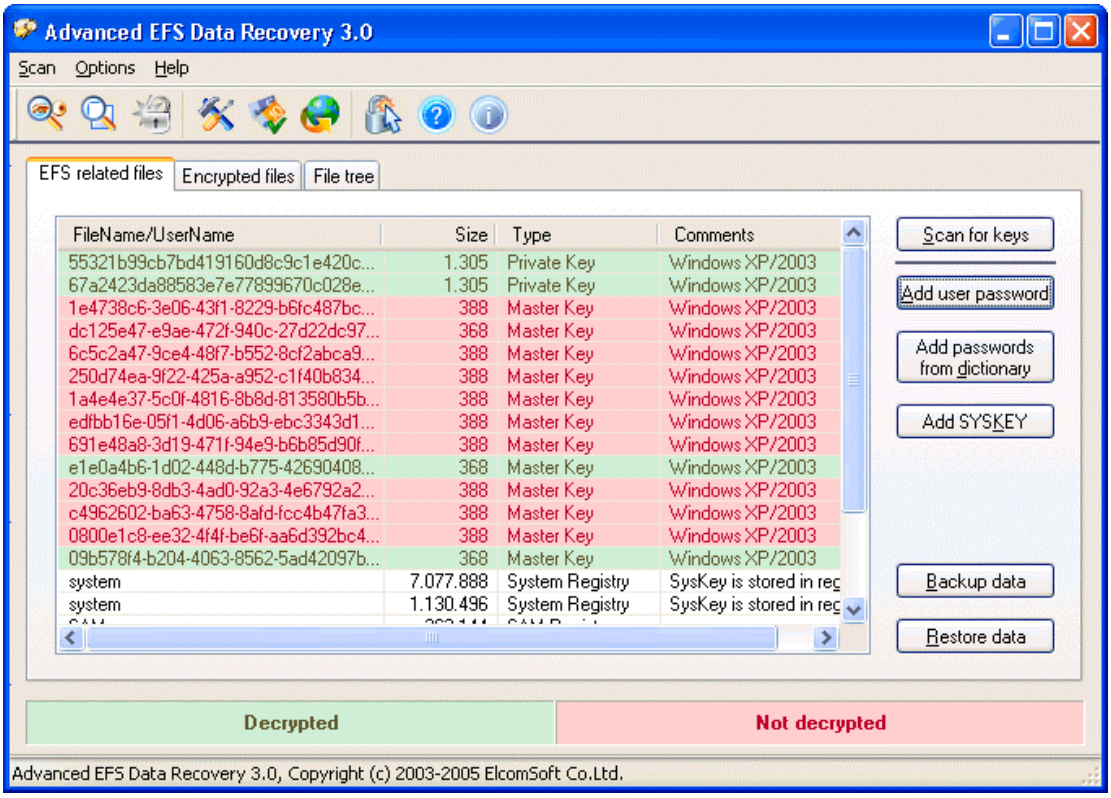



**Abbildung 9 Advanced EFS Data Recovery**

Grundsätzlich gilt:

Sobald ein Schlüssel auf dem verschlüsselten System liegt, ist das System unsicher.
Sobald der Schlüssel ohne Passwort dort ist, kann man nur noch hoffen, dass er nicht gefunden wird ;-)

### 4. Wir setzen das Root Passwort zurück

So und nun kommt wie angekündigt auch der praktische Teil!
Vorraussetzung dass das nachfolgende im Tutorial 1:1 angewendet werden kann ist, dass auf dem System der Bootmanager **Grub** oder **Lilo** installiert ist.
Dieser Weg sollte aber problemlos mit allen Linux Derivaten funktionieren.
Also wir starten den Computer und warten bis Grub erscheint...

Sobald wir in den Bootoptionen sind geben wir den Parameter **init=/bin/bash** ein und bestätigen danach mit der Eingabetaste.

[5] Elcomsoft – http://www.elcomsoft.com

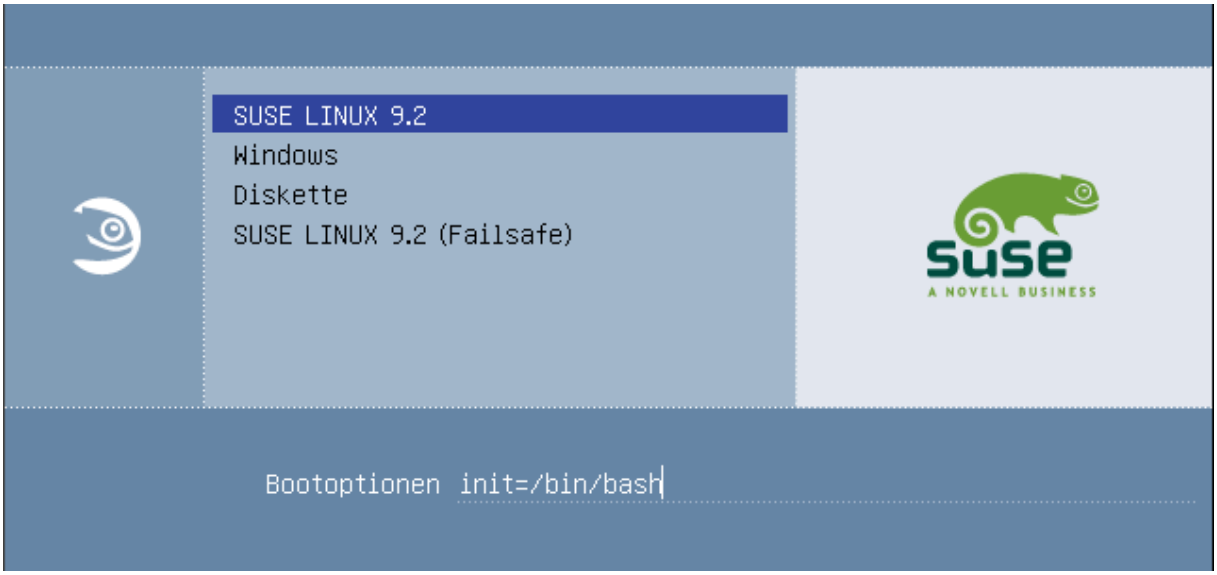



**Abbildung 10: Der Bootmanager Grub**

Mit diesem Parameter weisen wir den Kernel an, dass er statt der standardmässigen **/sbin/init** die **/bin/bash** laden soll.
**Bash** steht für **Bourne again Shell**. Man kann sie auch als wiedergeborene Shell ansehen (Born again Shell).
Die **Bash** ist voll kompatibel zur originalen **Bourne Shel**l (**sh**), ist jedoch im Funktionsumfang erheblich erweitert.

```
NET: Registered protocol family 8
NET: Registered protocol family 20
ACPI: (supports S0 S1 S5)
ACPI wakeup devices:
 USB
md: Autodetecting RAID arrays.
md: autorun ...
md: ... autorun DONE.
RAMDISK: Compressed image found at block 0
VFS: Mounted root (ext2 filesystem).
Starting udev
Creating devices
Loading kernel/fs/reiserfs/reiserfs.ko
Waiting for device /dev/hda7 to appear:  ok
rootfs:  major=3 minor=7 devn=775
ReiserFS: hda7: found reiserfs format "3.6" with standard journal
ReiserFS: hda7: using ordered data mode
reiserfs: using flush barriers
ReiserFS: hda7: journal params: device hda7, size 8192, journal first block 18, max trans
max batch 900, max commit age 30, max trans age 30
ReiserFS: hda7: checking transaction log (hda7)
reiserfs: disabling flush barriers on hda7
ReiserFS: hda7: Using r5 hash to sort names
VFS: Mounted root (reiserfs filesystem) readonly.
Trying to move old root to /initrd ... failed
Unmounting old root
Trying to free ramdisk memory ... okay
Freeing unused kernel memory: 212k freed
(none):/#
```

**Abbildung 11: Die Bourne again Shell**

Wir haben nun tatsächlich ohne Authentifizierung die Bourne again Shell vor uns.

Nun möchte ich mit dem altbekannten Befehl "**passwd root**" das Root Passwort zurücksetzen!

```
Freeing unused kernel memory: 212k freed
(none):/# passwd root
Changing password for root.
New password:
Bad password: too simple
Re-enter new password:
Cannot lock password file: already locked.
Error: Password NOT changed.
passwd: Authentication token lock busy
(none):/#
```

Als erstes motzt mich die Kiste an, dass mein Passwort zu simpel sei! Klar ich habe auch **12345** für das neue root Passwort angegeben ;-)

**Abbildung 12: Versuch Fehlgeschlagen!**

Zu meiner Enttäuschung sehe ich auch eine nervige Error Message:
**"Cannot lock password file: already locked"**

☞ **Password NOT changed.**

Verärgert stelle ich fest, dass ich nur Leserechte auf dem Datenträger habe.
Aber wir geben noch nicht auf!

Die nächste Überlegung ist den Datenträger neu zu **mounten** und zwar mit **Lese** und **Schreibrechten**

Wir mounten die Partition erneut mit folgender Befehlssyntax:

**„mount –n –o remount,rw /"**

Alle momentan gemounteten Dateisysteme sind in der Datei /**etc**/**mtab** gespeichert.
Wir können dies Problemlos mit dem Befehl **cat** /**etc**/**mtab** überprüfen.
Mit dem Parameter **–n mounte** ich den Datenträger ohne Schreibrechte nach /**etc**/**mtab**. Der Parameter –o steht für options, damit ich das **remount** und das **rw** für Lese und Schreibrechte Problemlos anhängen kann.

```
A device can be given by name, say /dev/hda1 or /dev/cdrom,
or by label, using  -L label  or by uuid, using  -U uuid .
Other options: [-nfFrsvw] [-o options] [-p passwdfd].
For many more details, say  man 8 mount .
(none):/# mount -n -o remount,rw /
(none):/# _
```

**Abbildung 13: Eingabe des Mount Parameters**

Danach haben wir Lese und Schreibrechte auf dem System und können dem root account ein neues Passwort setzen.

Also erneut **passwd root** eingeben.

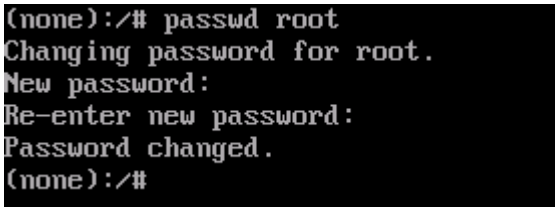



**Abbildung 14: Versuch erfolgreich!**

Wir sehen keine Fehlermeldung mehr sondern **Password changed** ;-)
Wir haben es geschafft! Das root Passwort wurde erfolgreich zurückgesetzt ohne, dass wir uns an das /**etc**/**passwd** oder /**etc**/**shadow** File heranmachen mussten.

So nun wäre ich am Ende meines kleinen Tutorials angelangt!
Ich hoffe ihr konntet etwas lernen und hattet Spass beim lesen ;-)

Bis zum nächsten Manual...

© 6/2005 by Daniel Müller
Last Update 12/2005

http://www.daniel85.ch.vu

Besten Dank an Herrn Gwerder, der mir immer wieder sehr gute Tipps zur Verbesserung meiner Manuals gibt!

## 5. Mini Glossar

| | |
|---|---|
| 1. Algorithmus | Unter einem Algorithmus versteht man eine mehr oder weniger genau definierte Handlungsvorschrift zur Lösung eines Problems oder einer bestimmten Art von Problemen. |
| 2. Bash | Bash steht für Bourne again Shell. Die Bash ist die Standardshell unter Linux und wurde auf fast alle Unix Systeme portiert |
| 3. Bios | Basic Input Output System. Eine Sammlung von im Computer eingebauten Softwarecodes, die für die Datenweiterleitung von einem Teil des Computers in einen anderen zuständig sind. |
| 4. DOS | DOS ist die Abkürzung für „Disk Operating System“. Es handelt sich dabei um ein textbasierendes Betriebssystem. |
| 5. EFS | Abkürzung für Encryption File System. System zum verschlüsselten Speichern von Daten, das zum Dateisystem NTFS gehört und unter Windows 2000 und XP verwendet wird. |
| 6. Grub | Grub steht für Grand Unified Bootloader. Gängiger Bootloader bei Linuxsystemen. |
| 7. Hashwert | Begriff aus der Informatik, der eine Zahl/Zeichenfolge bezeichnet, die aus einem Dokument oder einem Programm errechnet wird. Vereinfacht ist ein Hash auch vergleichbar mit einer Prüfsumme. |
| 8. Kernel | Kern des Betriebssystems bei Linuxsystemen, der alle Grundle genden Funktionen für den Zugriff auf die Hardware bereitstellt. |
| 9. LILO | Steht für Linux Loader. Ebenfalls ein Boormanager, der zu Linux gehört. |
| 10. mounten | Mit mounten wird das Einhängen eines Laufwerkes oder eines Dateisystems in ein bereits laufendes Rechnersystem bezeichnet. |
| 11. POST | Der Selbsttest nach dem Einschalten, ein Vorgang den der PC beim Hochfahren durchläuft um zu prüfen, ob Grundlegende Komponenten des PCs funktionsfähig sind. |
| 12.Root | Englisch für Wurzel. Systemaccount des Systemadministrators bei Unix Derivaten |
| 13. Shell | Textbasierende Benutzeroberfläche eines Betriebssystems. |