Bedrohung der Systemsicherheit durch Rootkits

Wilhelm Dolle, Berlin, 22. Oktober 2005

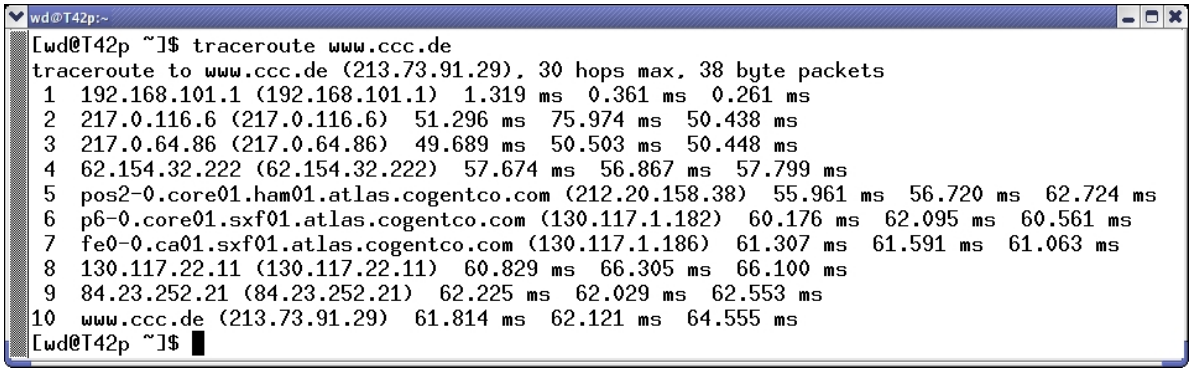
Agenda

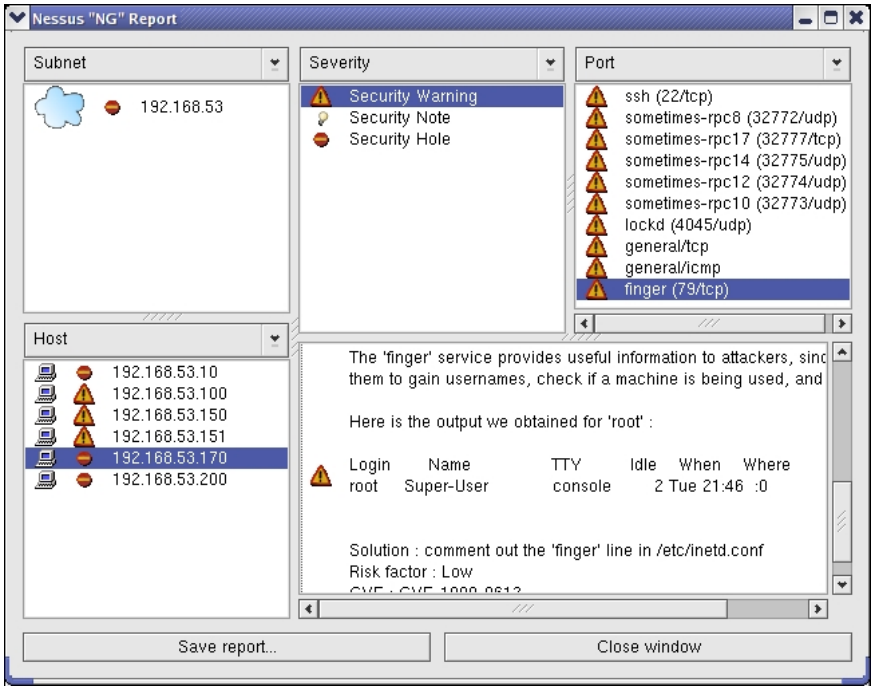
- **Was ist ein Rootkit?**
- **Klassifizierung und Möglichkeiten von Rootkits**
- **Rootkits im Einsatz**
- **Rootkits aufspüren**
- **Werkzeuge zum Aufspüren von Rootkits**

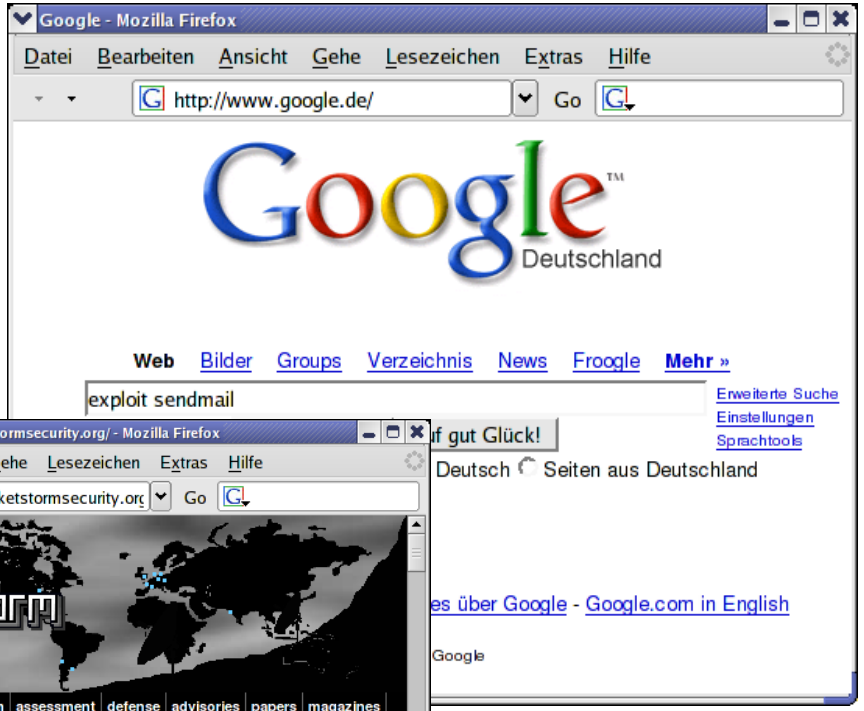
Typischer (gezielter) Einbruch übers Netz

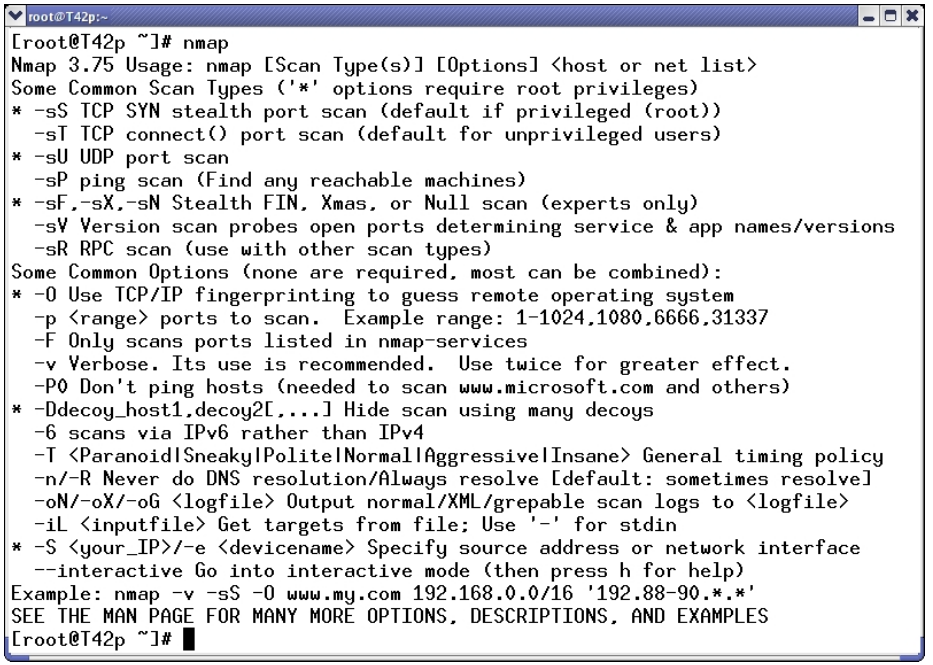
- **Footprinting** (allgemeine Informationen über Ziel sammeln)
  - öffentliche Datenbanken, Google, Unternehmens-Webseiten, DNS-Einträge, Internetanbindung per traceroute, ...
- **Protokoll- und Portscans / Enumeration / Schwachstellenanalyse**
  - Betriebssystemerkennung, Bannergrabbing, Verwundbarkeitsscans
- **Ausnutzen einer Sicherheitslücke**
  - Exploits (Buffer Overflow, ...) anwenden
- **Verstecken und Festsetzen**
  - Hintertüren, Sniffer, Keylogger, ...
- **Einbruchsspuren verwischen**
- **System missbrauchen**
  - Plattform für weitere Angriffe, Botnetze, Datenspionage, ...

Werkzeuge für die Informationsbeschaffung









Was ist ein Rootkit?

- „Ein Rootkit ist ein Satz von Programmen der dauerhafte und nicht aufzuspürende Gegenwart auf einem Computer erlaubt.“ (Hoglund, Butler)
- Der Fokus von Rootkits ist die Heimlichkeit (durchaus auch für legale Zwecke)
- Weitere Funktionen
  - Hintertüren
  - Fernsteuerung von Rechnern
  - Spyware (Netzwerk-Sniffer, Key-Logger, ...)

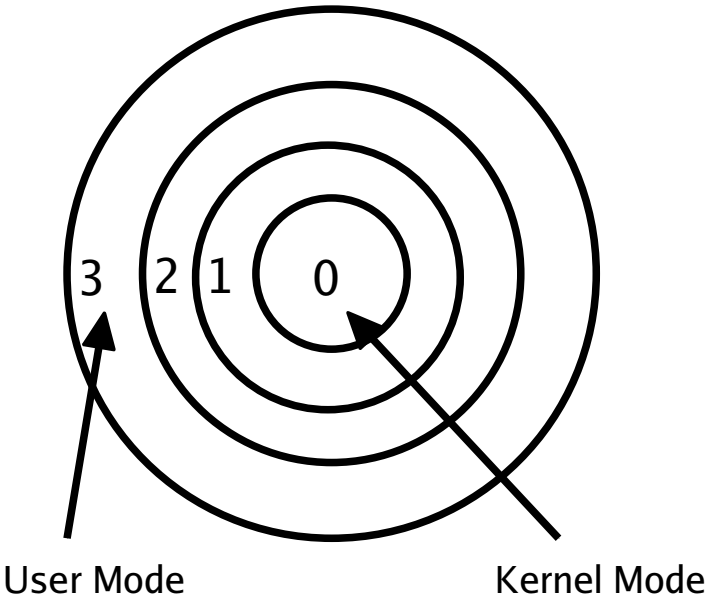
- Was ist ein Rootkit **nicht**?
  - Angriffswerkzeug oder Exploit (es kann aber Exploits enthalten)
  - Wurm oder Virus (Rootkit-Technik kann aber in diese integriert werden)

Klassifikation

- **Dateibasierte Rootkits** (User Mode Rootkits)
  - Manipulieren Systemprogramme oder Bibliotheken im Dateisystem
- **Kernelbasierte Rootkits** (Kernel Mode Rootkits)
  - Kommen durch Module (LKM) oder direkte Manipulation im RAM oder auf der Festplatte in den Kernel
  - Manipulieren Datenstrukturen bzw. Übergabewerte direkt im Kernel
- **Memory Based Rootkits**
  - Befinden sich nur im RAM und überstehen einen Reboot nicht
- **Persistente Rootkits**
  - Nisten sich auf der Festplatte ein und werden nach Reboot wieder aktiviert

Unterschied zwischen User und Kernel Mode

- Intel x86-Familie nutzt Ringe 0 bis 3 für Zugriffssteuerung
- Ring 0 hat die höchsten, Rind 3 die niedrigsten Privilegien zum Zugriff
- Viele Betriebssysteme (u.a. Linux, Windows) nutzen nur 0 (Kernel Mode) und 3 (User Mode)



- Zugriff auf Ringe mit kleinerer Nummer ist (bis wenige auf Ausnahmen, z.B. Laden von Treibern) verboten
- Kernel Mode (OS-Kernel) hat unbeschränkten Zugriff auf das ganze System
- Ring 3 enthält alle anderen Programme (**auch die mit Admin-Rechten!**)
- Rootkits im Kernel können beliebig Sicherheitssoftware im User-Mode manipulieren oder beenden, sie bestimmen welche Daten die Software sieht und bekommen z.B. Netzwerkpakete vor einem Paketfilter (stealth backdoor)

Für Kernel Rootkits interessante Systemaufrufe

- **open()**
  lesender Zugriff ergibt Original, ausführender Zugriff ergibt modifizierte Version

- **getdents(), mkdir(), chdir(), rmdir()**
  Verstecken von Dateien und Verzeichnissen

- **execve(), clone(), fork()**
  Programme verstecken und Vererbung an Kindprozesse

- **stat()**
  Manipulation von Dateieigenschaften

- **ioctl()**
  Device-Kontrolle, z.B. kein promiscuous-Bit (Sniffer) anzeigen

Historie von Rootkits

- Ende 80er: Manipulieren von Logfiles
- 1989: Phrack-Magazin-Artikel zum Umgehen von Unix-Übrwachung
- 1994: CERT-Hinweis auf Sammlung von Programmen unter SunOS („rootkits“)
- 1996: erste Linux-Rootkits
- 1997: Phrack-Magazin-Artikel zu LKM-Rootkits unter Linux (später auch UNIX)
- 1998: direktes Patchen des Kernels im RAM ohne LKM (Silvio Cesare)
- 1999: LKM-Rootkits: knark (speziell zur Täuschung von Tripwire), adore
- 1999: erstes Windows Rootkit (NT-Rootkit)
- 2000: t0rnkit v8 libproc library Rootkit
- 2001: KIS und SucKIT manipulieren Kernel direkt im RAM (Technik aus 1998)
- 2002: erste Rootkits mit Sniffer-Backdoors

Beispiel User Mode Rootkit: t0rnkit

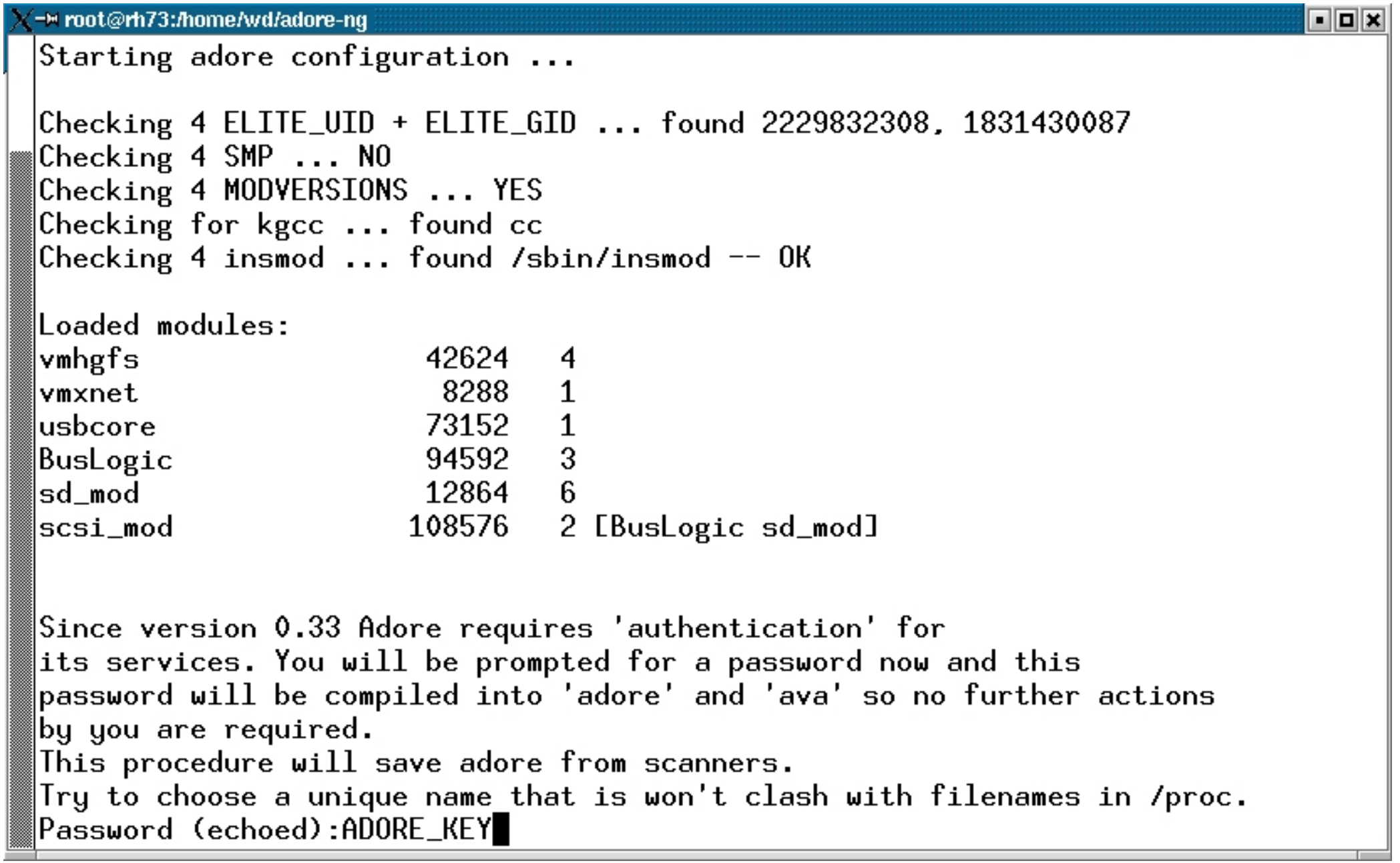
- syslogd anhalten
- Hash-Wert des Passwortes der Hintertür in /etc/ttyhash ablegen
- Trojanisierter SSH-Dienst unter /usr/sbin/nscd ablegen
- Start als „# Name Server Cache Daemon ...“ in der Startdatei /etc/rc.d/sysinit (Defaultport 47017)
- Konfigurationsdateien unter /usr/info/.t0rn ablegen
- Austauschen der folgenden Systemprogramme (Zeitstempel und Größen werden zurückgesetzt): login, ls, netstat, ps, ifconfig, top, du, find
- in.fingerd öffnet eine Shell am Defaultport 2555
- Verschiedene Binarys (z.B. Sniffer) unter /usr/src/.puta und /dev/.lib ablegen
- Aktivieren von telnet, rsh und finger in /etc/initd.conf
- syslogd neustarten

Beispiel Kernel Mode Rootkit: adore-ng

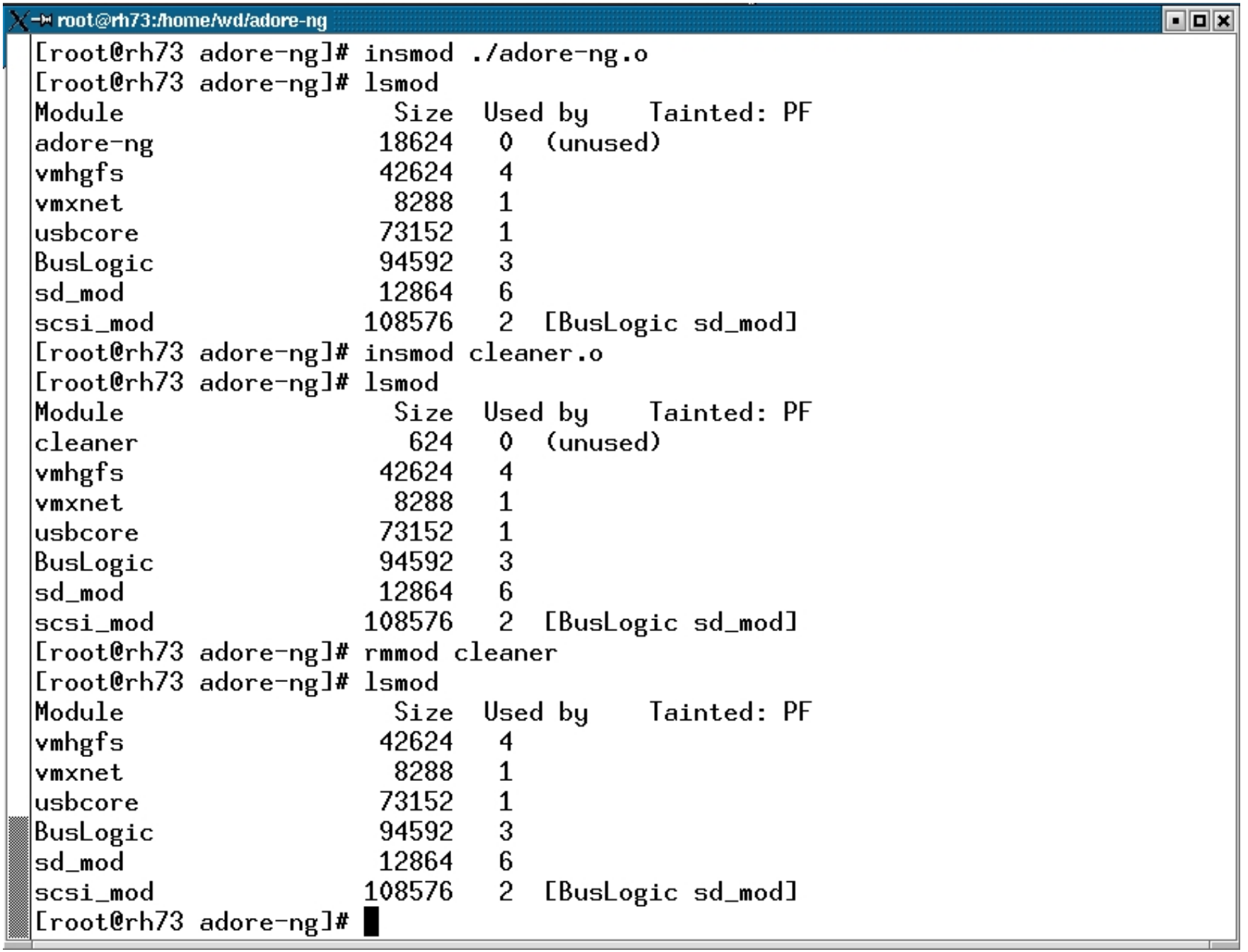
- Entwickelt von Team Teso für Linux Kernel 2.2, inzwischen auf 2.4 und als adore-ng auch auf 2.6 portiert
- Linux Kernelmodul, Cleaner-Modul, Kommandozeilenwerkzeug

- Features
  - Verstecken von Dateien und Verzeichnissen
  - Verstecken von Prozessen
  - Verstecken von Netzwerkverbindungen
  - Hintertür um die Rechte auf dem System zu erweitern
  - Filtern von Logfile-Einträgen
  - ...

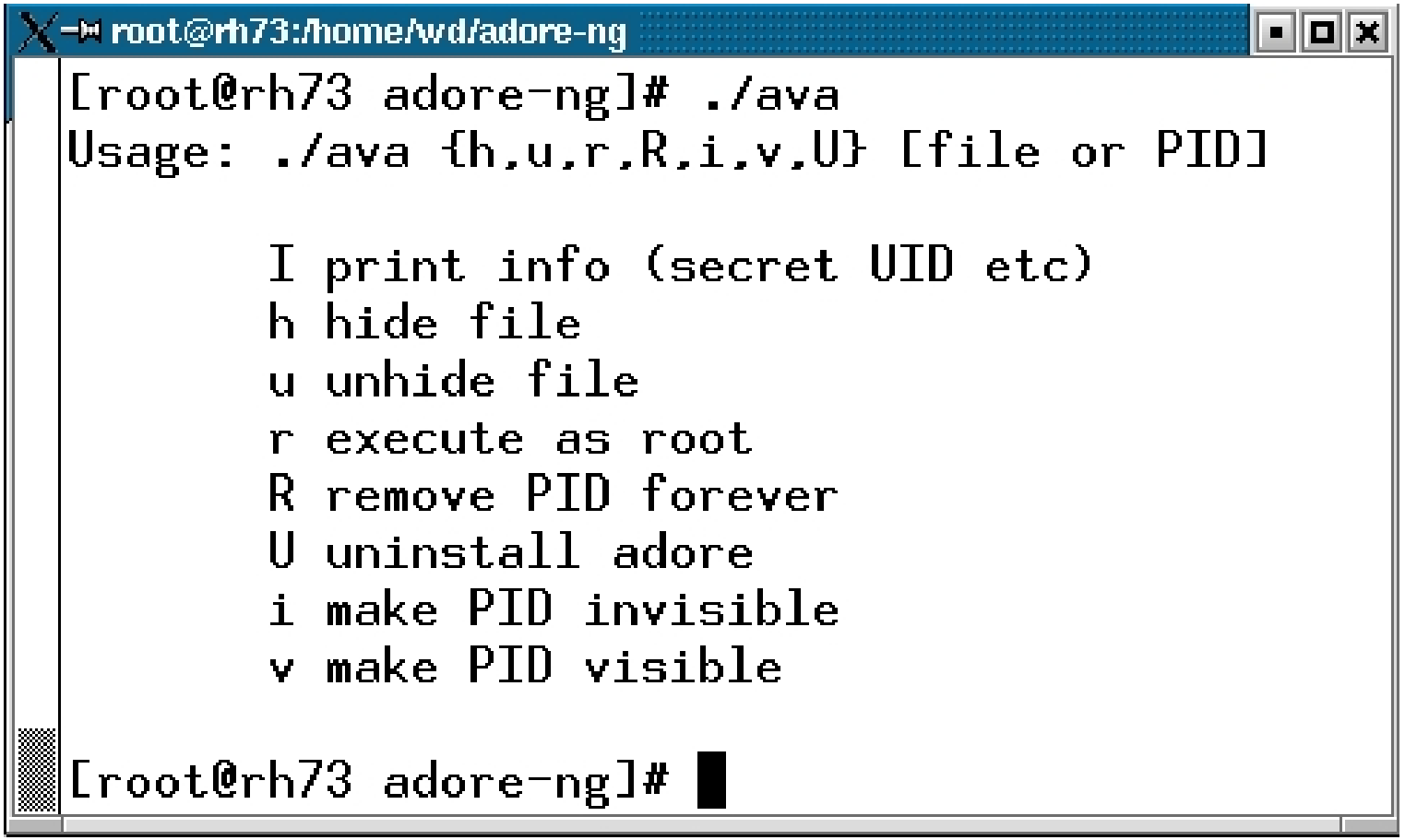
adore-ng - Konfiguration



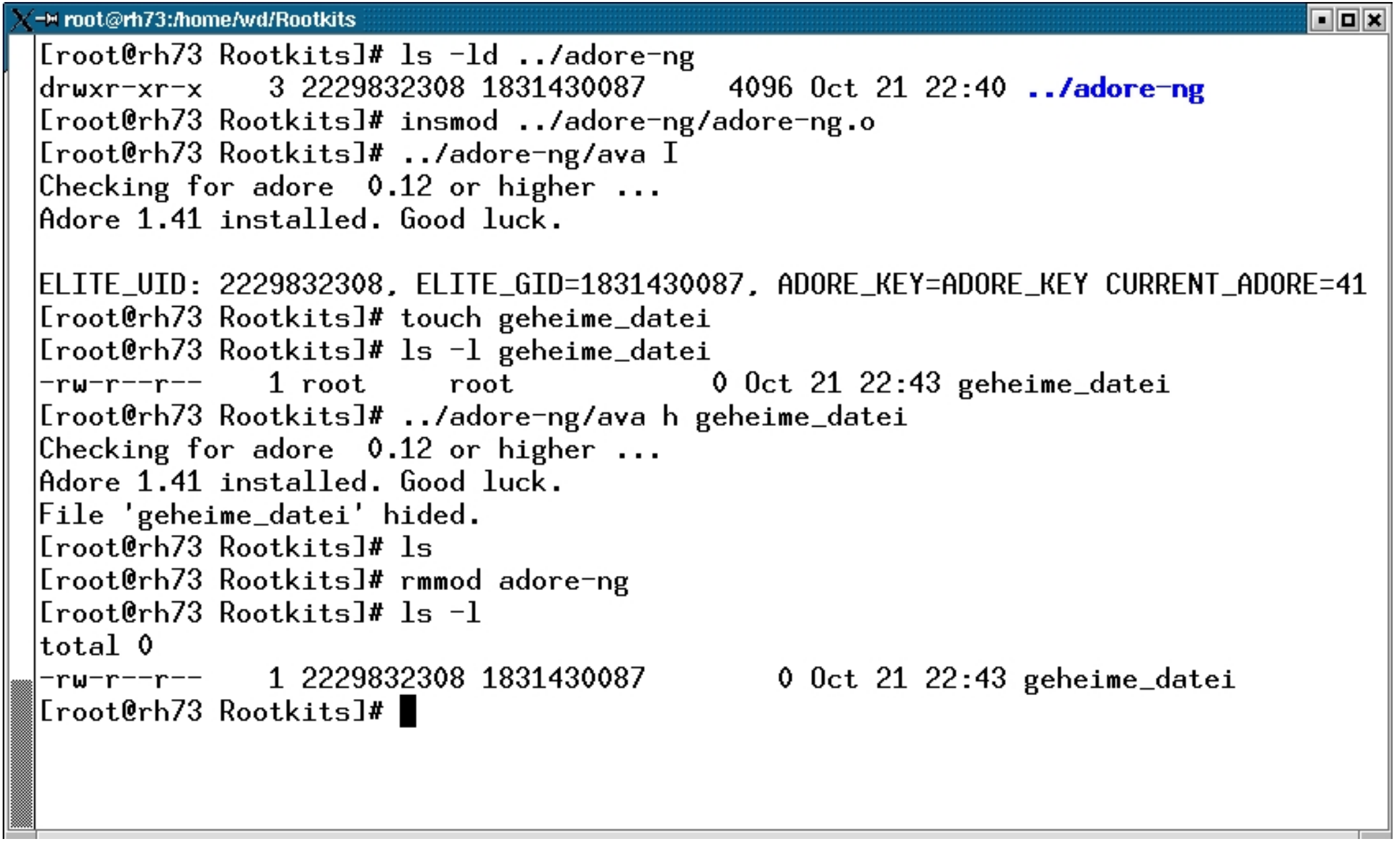
adore-ng – Starten und Verstecken



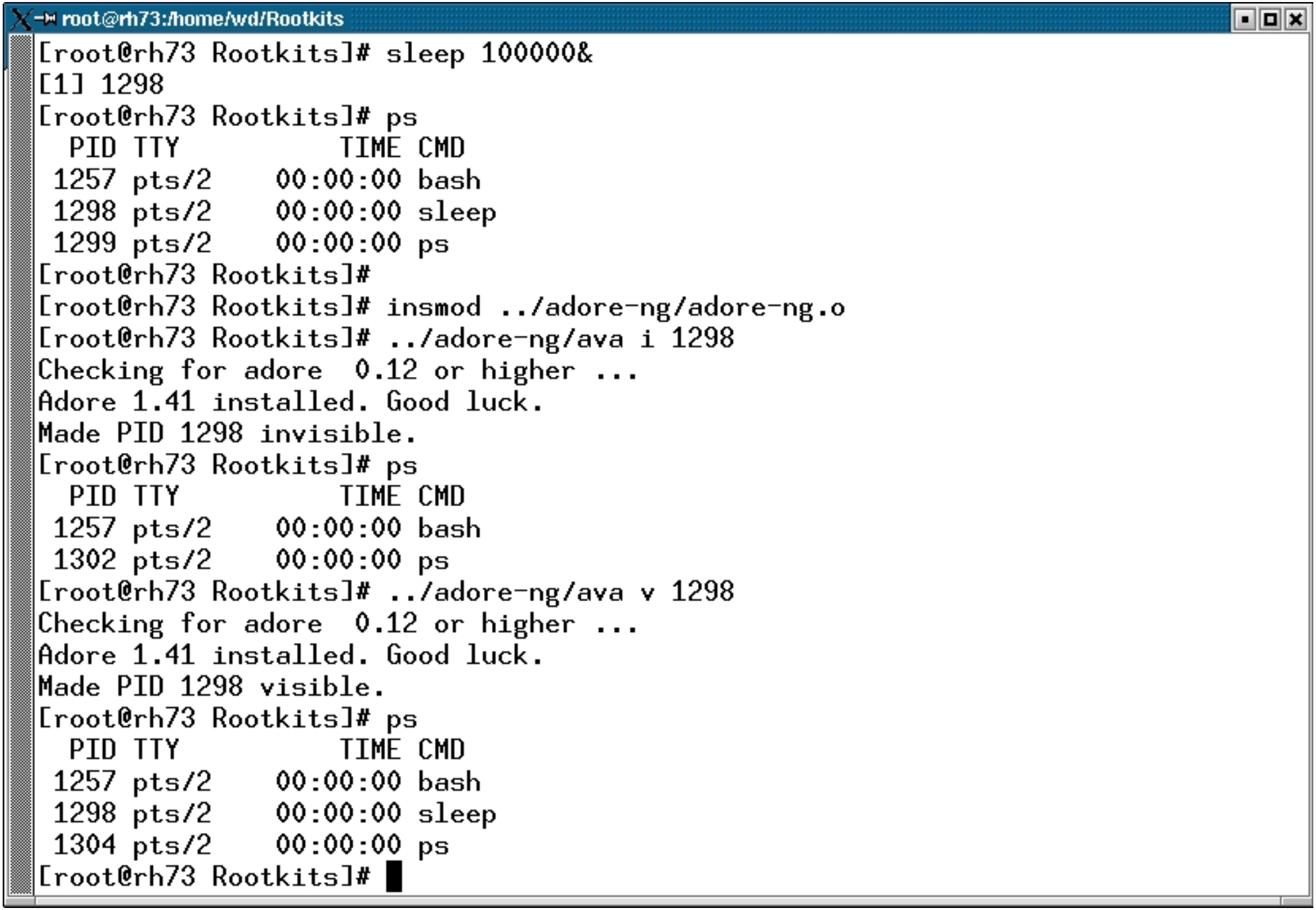
adore-ng – Kommandozeilentool ava



adore-ng – Dateien / Verzeichnisse verstecken



adore-ng – Prozesse verstecken



Rootkits aufspüren I

- Startscripte prüfen
- Oft trojanisierte Programme prüfen: ps, ls, find, ifconfig, netstat, du, df, sshd, httpd, login, passwd, ...
- Verdächtige Rechte oder MAC-Times in /sbin oder /usr/sbin? (im Zweifel Binarys mit „strings“ ansehen)
- Dateien mit Link Count kleiner 1 (lsof +L1)
- History der Shell ansehen (z.B. .bash_history)
- Normale Dateien im Device-Dateisystem (find /dev -type f)?
- Geladene Module prüfen
- Nach Interfaces im promiscuous-Mode suchen (Sniffer)
- Analyse der installierten Pakete mit „rpm --verify --all“ (Anzeige: S,5,T, ...)

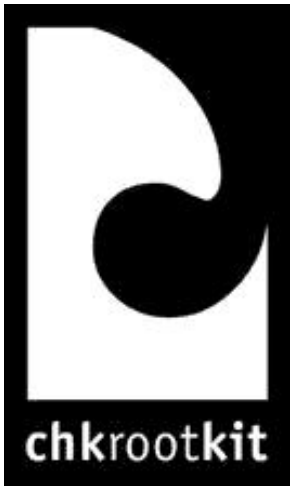
Rootkits aufspüren II

- Logfiles auf verdächtige Einträge durchsuchen (Neustart von Diensten war evtl. Buffer Overflow?)
- Portscan von außen mit einem internen Portscan oder netstat vergleichen (versteckte Port, findet aber nicht Sniffer/hidden Backdoors)
- Signaturen von bekannten Rootkits im Speicher und in Dateien suchen
- Durch PID-Test versuchen alle freien PID durch einen Testprozess zu belegen (nicht belegbare PIDs können ein Hinweis auf versteckte Prozesse sein)

- Cross-View-Based Rootkit-Detection: Vergleich der Sicht durch einen High-Level-Zugriff über die APIs mit einem Low-Level-Zugriff auf „Rohdaten“ (z.B. Lesen von einzelnen Sektoren von der Platte statt über das Dateisystem)

Werkzeuge zum Aufspüren: ChkRootKit

- www.chkrootkit.org, lokal ausführbares Script
- Erkennt sehr viele bekannte Rootkits
  (Stand Februar 2005, Version 0.45, ca. 60)
- Benutzt lokale Befehle, darum mit statisch
  gelinkten Binarys von CD benutzen




```
root@rh73:/home/wd/chkrootkit-0.45
Checking `asp'... not infected
Checking `bindshell'... not infected
Checking `lkm'... Warning: Adore LKM installed
ps: error: Thread display not implemented.
usage: ps -[Unix98 options]
       ps [BSD-style options]
       ps --[GNU-style long options]
       ps --help for a command summary
OooPS!
chkproc: Warning: Possible LKM Trojan installed
Checking `rexedcs'... not found
Checking `sniffer'... eth0: not promisc and no PF_PACKET sockets
Checking `w55808'... not infected
Checking `wted'... chkwtmp: nothing deleted
Checking `scalper'... not infected
Checking `slapper'... not infected
Checking `z2'... chklastlog: nothing deleted
Checking `chkutmp'... chkutmp: nothing deleted
[root@rh73 chkrootkit-0.45]#
```

Werkzeuge zum Aufspüren: Rootkit Hunter

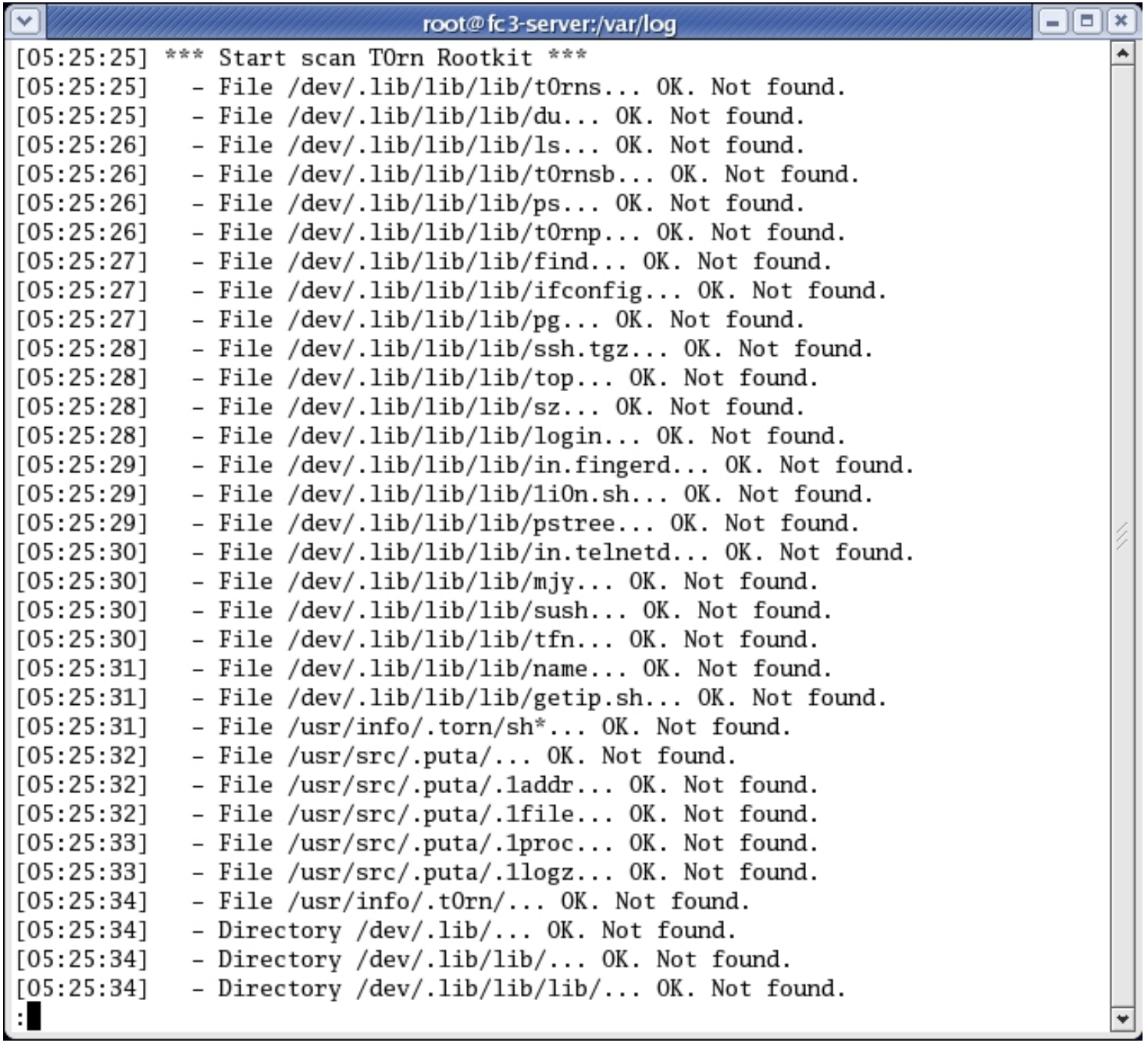
- www.rootkit.nl, aktuelle Version 1.2.7, Mai 2005
- Benutzt unter anderem die folgenden Methoden
  - MD5 Hash-Vergleich mit bekannten Werten
  - Suche nach typischen Dateien, Verzeichnissen und Ports die von Rootkits genutzt werden
  - Falsche Berechtigungen von Binärdateien
  - Suche nach verdächtigen Zeichenketten im LKM
  - Suche nach versteckten Dateien (in den falschen Verzeichnissen)
  - Betriebssystem abhängige Tests (bei Linux z.B. Vergleich der Prozesse über die Ausgabe von „ps“ und das parsen von /proc)
  - Optional scannen innerhalb von Plaintext und Binärdateien nach verdächtigen Zeichenketten

Rootkit Hunter im Einsatz I

```
root@fc3-server:/home/wd/Rootkits/rkhunter
[root@fc3-server rkhunter]# rkhunter -c --createlogfile


Rootkit Hunter 1.2.7 is running

Determining OS... Ready


Checking binaries
* Selftests
     Strings (command)                                [ OK ]


* System tools
Info: prelinked files found

   /bin/cat
   /bin/chmod
   /bin/chown
   /bin/dmesg
   /bin/egrep
   /bin/env
   /bin/fgrep
```

```
root@fc3-server:/home/wd/Rootkits/rkhunter
---------------------------- Scan results ----------------------------

MD5
MD5 compared: 72
Incorrect MD5 checksums: 0

File scan
Scanned files: 342
Possible infected files: 0

Application scan
Vulnerable applications: 2

Scanning took 2093 seconds
Scan results written to logfile (/var/log/rkhunter.log)

------------------------------------------------------------------------

Do you have some problems, undetected rootkits, false positives, ideas
or suggestions?
Please e-mail me by filling in the contact form (@http://www.rootkit.nl)

------------------------------------------------------------------------
[root@fc3-server rkhunter]#
```

Rootkit Hunter im Einsatz II



Entfernen von Rootkits

- Anhand der gefundenen Spuren im Internet nach Informationen und Hinweisen zur Entfernung des Rootkits suchen
  - Module entfernen
  - Manipulierte Dateien durch Originale ersetzen
  - Startscripte säubern

- **Wann immer möglich System komplett neu aufsetzen!**

Q & A

- **Vielen Dank für die Aufmerksamkeit!**
- **Folien unter www.dolle.net**
- **„Rootkits“ (Windows);**
  Hoglund, Butler; Addison-Wesley 2005