NewbieNet



Sicherheit

Absichern einer Dial−Up−Internetverbindung unter Linux

Autor: Karsten Kruse

Inhalt

- Vorwort
- Grundsätzliches
- Dienste in inetd abschalten
- TCP−Wrapper
- Andere Dämonen
- Paketfilter
    - Iptablespaketfilter (Kernel 2.4.x)
- Nachwort

Vorwort

Hier sind ein paar kleine Kniffe wie du deine Dial−Up−Internetverbindung sicherer machen kannst. Die Tricks sind recht simpel, aber wirkungsvoll genug um auf eine komplexe Firewall verzichten zu können. Einen wirklich ausführlichen Einblick gibt der Linux Administrators Security Guide.

Grundsätzlich

Ein Linuxrechner bietet oft von Haus aus einige Netzwerk−Dienste an, diese werden von den sogenannten Dämonen bereitgestellt. Ein sehr bekannter Netzwerkdienst ist WWW der oft vom Dämon Apache bereitgestellt wird. Weniger bekannt bei den Einsteigern sind die Dämonen telnetd, sshd oder auch identd. Normalerweise brauchen die meissten Dämonen nicht auf dem eigenen Rechner zu laufen. Auf andere möchte man aber dann doch lieber nicht verzichten. So kann es auf einem Linuxrechner immer gut sein sshd laufen zu lassen weil man sich eine verschlüsslte Verbindung zum Linuxrechner über das Netzwerk aufbauen kann. Also verzichten wir auf Telnetd (der dasgleiche anbietet, nur unverschlüsselt). Damit haben wir eine potentielle Angriffsfläche für einen Cracker geschlossen. Je weniger Dienste wir anbieten desto schwieriger ist es den Rechner anzugreifen. Also versuchen wir erstmal alle Dienste abzuschalten die wir nicht benötigen.

Dienste in inetd abschalten

Ein Linuxsystem hat oftmals einen speziellen Dämon namens inetd am laufen der auch als "Internet Superserver" bezeichnet wird. Er läuft immer und startet bei Bedarf andere Dämonen die gerade gebraucht werden. Der Vorteil liegt auf der Hand: Wenn inetd läuft und nur bei Bedarf z.b. pop3d, ftpd und talkd startet haben wir eben nur einen Dämon am laufen, statt drei. Das trägt mit dazu bei das die Systemlast gesenkt wird, hat aber auch den Nachteil das es ein paar Millisekunden länger dauert den Dienst bereitzustellen. Doch es gibt noch einen wichtigen Vorteil: Der inetd ermöglicht eine erweiterte Zugriffskontrolle durch TCP−Wrapper, dazu später mehr.

Welche Dienste vom inetd bereitgestellt werden hängt davon ab was in seiner Konfigurationsdatei steht die als inetd.conf im Verzeichnis /etc liegt. Öffnen wir die Datei mal mit einem Editor und sehen sie uns an, so könnte das aussehen:

```
ftp           stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd
```

In diesem Beispiel sind alle Einträge bis auf ftp auskommentiert und damit inaktiv, die Dämonen sind also nicht erreichbar. Der Dienst ftp ist also solange erreichbar wie der inetd läuft. Wenn du einen der Dienste in deiner /etc/inetd.conf nicht brauchst kommentiere ihn aus. Starte danach den inetd neu um die neue Konfigurationsdatei einzulesen: "rcinetd restart". Normalerweise ist es gut alles zu verbieten, in diesem Fall muss der inetd auch nicht laufen (wozu auch?).

TCP−Wrapper

Die Dienste aus inetd.conf können mit TCP−Wrappern gesichert werden. Damit kannst du den Zugriff auf den Dienst von IP−Adressen abhängig machen. Die Dateien mit denen das eingestellt wird heissen /etc/hosts.allow und /etc/hosts.deny. Die Ausgabe von "man 5 hosts_access" erklärt ausführlich wie man diese Dateien zu handhaben hat, ich beschränke mich deshalb auf eine ganz kurze Einführung.

Beginnen wir mit der Datei /etc/hosts.deny:

```
ALL: ALL
```

Das erste ALL betrifft alle Dienste aus /etc/inetd.conf. Das zweite ALL betrifft alle IPs die zu einem Dienst aus inetd.conf Verbindung aufnehmen wollen. Also sagt die Zeile ALL:ALL in /etc/hosts.deny das niemand die Erlaubnis hat sich mit irgendeinem Dienst zu verbinden. Das ist ziemlich streng und wir sollten das auflockern. Dazu benutzen wir die Datei /etc/hosts.allow:

```
ALL : 127.0.0.1
```

Dies erlaubt allen lokalen Benutzern und Prozessen auf alle Dienste in /etc/inetd.conf zuzugreifen. Jetzt wollen wir noch allen Rechnern im LAN (sofern vorhanden) Zugriff auf alle Dienste erlauben sofern wir unserem LAN vertrauen:

```
ALL : 127.0.0.1, LOCAL, 192.168.1.
```

Jetzt noch ftp erlauben, aber für keinen Host der auf .crackers.net endet. Ausserdem soll für jeden ftp−Zugriff noch ein Kommando ausgeführt werden. In diesem Fall wird versucht Userinformationen mit finger zu erhalten, das Ergebnis wird dann per Mail an den User Karsten gesendet.

```
ALL : 127.0.0.1, LOCAL, 192.168.1.
in.ftpd : ALL EXCEPT .crackers.net (/usr/sbin/safe_finger −l @%h | /bin/mail −s %d−%h karsten) &
```

So wird das dann auch mit den anderen Diensten gemacht die man erlauben will.

Andere Dämonen

Nicht alle Dienste werden von inetd bereitgestellt, beispiele sind sendmail oder apache (obwohl das auch gehen würde, das Starten dauert nur zu lange). Also ist es egal wie streng deine Sicherheitsvorgaben für die TCP−Wrapper sind, du musst dir für diese Dienste etwas anderes einfallen lassen.

Als erstes solltest du feststellen welche Dienste gestattet werden. Unter SuSE−Linux findest du das so heraus:

```
karsten@tecneeq:~ >  grep START /etc/rc.config
```

Und so für RedHat−Linux:

```
karsten@tecneeq:~ >  chkconfig −−list | grep :on | awk '{ print $1 }'
```

Resultat ist eine Liste von Diensten die beim Booten gestartet werden. Nicht alles Dienste sind Serverdienste die wir beachten müssen, cron oder atd sind nicht interessant für uns. Doch einige Dienste sind Schützenswert, z.b. inetd (den wollen wir ja sowieso haben), named (Nameserver), lpd (Line−Printer−Dämon), sendmail (zum Mail senden), sshd (SecureShell). Sollte bei dir ein Dienst dabei sein den du nicht brauchst schalte ihn ab, das Handbuch deiner Distribution sagt dir wie es geht. Du kannst dir auch mit folgendem Kommando anschauen welche Dienste auf eine TCP−Verbindung horchen:

```
karsten@tecneeq:~ > netstat −ant
Aktive Internetverbindungen (Server und stehende Verbindungen)
Proto Recv−Q Send−Q Local Address        Foreign Address      State
tcp        0      0 0.0.0.0:32768        0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:513          0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:514          0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:32771        0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:869          0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:3306         0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:111          0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:6000         0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:80           0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:21           0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:119          0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:25           0.0.0.0:*            LISTEN
tcp        0      0 0.0.0.0:7869         0.0.0.0:*            LISTEN
tcp        0      0 192.168.1.20:32937   192.168.1.99:22      VERBUNDEN
tcp        0      0 192.168.1.20:32936   192.168.1.99:22      VERBUNDEN
tcp        0      0 192.168.1.20:32790   192.168.1.99:22      VERBUNDEN
tcp        0      0 192.168.1.20:6000    192.168.1.99:2609    VERBUNDEN
tcp        0      0 :::22                :::*                 LISTEN
```

Wenn du jetzt noch wissen willst welcher Port zu welchem Dienst gehört kannst du das mit Hilfe von /etc/services herausfinden. Hier ein Beispiel:

```
karsten@tecneeq:~ > grep 119 /etc/services
nntp            119/tcp                 # Network News Transfer Protocol
nntp            119/udp                 # Network News Transfer Protocol
```

Paketfilter mit Iptables

Wenn du ein "echter" Newbie bist wirst du deine überflüssigen Dienste abschalten und bist fertig. Warum? Weil Dienste die nicht gestartet sind nicht angreifbar sind, hast du also keine Dienste laufen gibt es auch keine Angriffsfläche. Die Dienste die mit inetd gestartet werden haben wir schon durch TCP−Wrapper gesichert, alles andere ist hoffentlich aus. Wenn ja bist du fertig, wenn nein lies weiter.

Doch wenn du ein "unechter" Newbie bist, so wie ich ;), dann willst du möglicherweise einen Server aufsetzen der mehrere Dienste anbietet die nicht über inetd gestartet werden aber trotzdem nicht vom Internet aus anzugreifen sind. Hier kommt die Firewall (eigentlich nur ein Paketfilter, wir sagen aber Firewall aus Gewohnheit) ins Spiel. Sinn der Firewall ist für uns das Blocken von Verbindungsanfragen aus dem Internet auf Dienste auf unserem Server.

Der Linuxkernel hat die Fähigkeit Pakete zu filtern. Der Kernel kann sich jedes Paket anschauen und nach von uns aufgestellten Regeln das Paket durchlassen, verwerfen, weiterleiten oder mit einer bestimmten Nachricht

beantworten, das ganze abhängig von der Paketart, der Quelle, des Ziels und einigen anderen Attributen. Für eine komplette Beschreibung solltest du die Manpage zu "iptables" und "ipchains" lesen. Es gibt für die verschiedenen Kernelversionen verschiedene "Frontends" um die Pakete zu steuern. Für die Kernels ab Version 2.2.x benutzt man Ipchains, für die Kernels ab Version 2.4.x Iptables. Du kannst rausfinden welche Kernelversion du hast indem du dieses Kommando eingibst:

```
karsten@tecneeq:~ > uname −sr
Linux 2.4.7
```

Iptablespaketfilter (Kernel 2.4.x)

Ich arbeite mit Iptables und werde jetzt ein kleines Firewallscript zeigen das du direkt benutzen kannst. Es ist nur ein Anfang, aber es sollte ausreichen um dir das Prinzip zu zeigen. Danach kannst du deine eigenen Regeln einbauen um die Firewall an deine Erfordernisse anzupassen (obwohl das meistens nicht nötig ist). Erstelle dir eine Datei "minifire" und trage das folgende ein:

```
#!/bin/sh
## Einstellungen
DIAL_OUT=ippp0
ALLOW_PORTS_TCP="21 22 80"
ALLOW_PORTS_UDP="21 22 80"
REJECT_PORTS_TCP="113"
REJECT_PORTS_UDP="113"
ALLOW_MASQUERADING=ja
ANTI_SPOOFING=ja
SYNCOOKIE=ja
ALLOW_FORWARD=$ALLOW_MASQUERADING
ALLOW_DYNIP=$ALLOW_MASQUERADING
ALLOW_PING=ja
## Ende Einstellungen

flush_chains()
{
   ## loesche alte tables und chains
   iptables −F
   iptables −X
}
start_firewall()
{
   ## lade module
   modprobe ip_tables
   modprobe iptable_nat
   modprobe ip_conntrack_ftp
   modprobe ip_nat_ftp

   ## starte anti−ip−spoofing / source−adress ueberprüfung
   if [ $ANTI_SPOOFING = ja ]; then
   echo −e "\tStarte Anti IP Spoofing"
   for FILTER in /proc/sys/net/ipv4/conf/*/rp_filter; do
       echo 1 > $FILTER
   done
   # logge gespoofte pakete, source routed pakete und redirect pakete
   for f in /proc/sys/net/ipv4/conf/*/log_martians; do
       echo 1 > $f
   done
   fi

   ## erlaube SYN cookie schutz
   if [ $SYNCOOKIE = ja ]; then
   echo −e "\tStarte SYN Cookie Schutz"
   echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

```
   fi

   ## erlaube forwarding
   if [ $ALLOW_FORWARD = ja ]; then
   echo −e "\tStarte Forwarding"
   echo 1 > /proc/sys/net/ipv4/ip_forward
   fi

   ## erlaube dyn_ip−patch
   if [ $ALLOW_DYNIP = ja ]; then
   echo −e "\tFIX ME"
   fi

   ## erstelle regeln um alle verbindungen von $DIAL_OUT zu blocken
   iptables −N block
   iptables −A block −m state −−state ESTABLISHED,RELATED −j ACCEPT
   iptables −A block −m state −−state NEW −i ! $DIAL_OUT −j ACCEPT
   iptables −A block −j DROP

   ## erlaube services
   for PORT in $ALLOW_PORTS_TCP ; do
       iptables −A INPUT −p tcp −−dport $PORT −j ACCEPT
   done
   for PORT in $ALLOW_PORTS_UDP ; do
       iptables −A INPUT −p udp −−dport $PORT −j ACCEPT
   done

   ## diese services werden ausdrücklich abgelehnt und nicht verworfen
   for PORT in $REJECT_PORTS_TCP ; do
       iptables −A INPUT −i $DIAL_OUT −p tcp −−dport $PORT −j REJECT
   done
   for PORT in $REJECT_PORTS_UDP ; do
       iptables −A INPUT −i $DIAL_OUT −p udp −−dport $PORT −j REJECT
   done

   ## erlaube ping
   if [ $ALLOW_PING = ja ]; then
   echo −e "\tErlaube Ping"
   iptables −A INPUT −p icmp −j ACCEPT
   fi

   ##
   iptables −A INPUT −j block
   iptables −A FORWARD −j ACCEPT

   ## starte masquerading
   if [ $ALLOW_MASQUERADING = ja ]; then
   echo −e "\tMasquerading"
   iptables −t nat −A POSTROUTING −o $DIAL_OUT −j MASQUERADE
   fi
}
usage ()
{
echo "
$0 − eine Minifirewall für Newbies :)
Benutzung:
   $0 [option]

Optionen:
   −−start    −> Startet die Firewall
   −−stop     −> Stoppt die Firewall
   −−status   −> Zeigt die Regeln der Firewall
   −−restart  −> Startet die Firewall neu
   −h −−help  −> Gibt diese Hilfe aus
```

```
Sonstiges:
   Vor dem ersten Start sollten die Variablen am Anfang
   des Scriptes bearbeitet werden.
"
}
case "$1" in

   --start)    flush_chains
               start_firewall
               echo "Firewall aktiviert"
               exit ;;

   --stop)     flush_chains
               echo "Firewall deaktiviert"
               exit ;;

   --restart)  flush_chains
               echo "Firewall deaktiviert"
               start_firewall
               echo "Firewall aktiviert"
               exit ;;

   --status)   iptables -L -n
               exit ;;

   -h|--help)  usage
               exit ;;

   *)          usage
               exit 1 ;;
esac
exit
```

Mit "chmod 700 minifire" geben wir unserer Firewall die richtigen Rechte und kopieren sie nach /usr/local/sbin. Jetzt schreiben wir in die Datei /etc/init.d/boot.local noch folgende Zeile um die Firewall beim Booten zu starten:

```
/usr/local/sbin/minifire --start
```

Jetzt müssen nur noch die Variablen am Anfang des Skriptes bearbeitet werden um die Firewall anzupassen. Danach kann die Firewall sofort gestartet werden mit "minifire --start". Sie ist jetzt aktiv. Als nächstes sollten wir Testen ob die Firewall auch funktioniert. Dazu rufen wir diese Webseite auf und machen den "Selbsttest": Zum Selbsttest. Fertig.

Nachwort

Noch ein Hinweis: Die Firewall schützt nur nach "aussen", gegen Angriffe von innen ist sie Machtlos. Auch sollten die Dienste immer in der neuesten Version installiert sein denn was bringt die beste Firewall wenn der Dienst an sich unsicher ist.

Kommentare, Wünsche und Anträge an Karsten Kruse.

[PDF - Text]