

# Little Secure Firewall System

Firewall mit eigener Linux-Distribution

Hayati Aygün (Dipl. Informatiker)  
h\_ayguen@web.de

Cornelius Wasmund (IT-Berater)  
Cornelius.Wasmund@pf-lug.de

# Firewall?!

- TCP/IP Verbindung ins Internet mit folgenden Diensten oft unabdingbar:
  - World Wide Web (HTTP)
  - eMail (IMAP bzw. SMTP/POP3)
- Schutz vor Hackern aus der Wösen Weiten Welt:
  - Lahmlegen/Misbrauch der eigenen Rechner: DoS, SPAM Versand, verteilte Angriffe, ..
  - Ausspähen von vertraulichen Daten
- Schutz vor eigenen Mitarbeitern:
  - Versand von vertraulichen Daten

# Risikominimierung

- Dienste bergen höheres Risiko, da:
  - (oft) öffentlich zugänglich – auch für Hacker
  - Rund um die Uhr aktiv
  - Bei Client Anwendungen steuert Anwender zu welchen Rechnern Verbindung aufgebaut wird
- Schutz vor DoS / Hackerangriffen
  - Aktuelle Sicherheitspatches aufspielen
  - Einsatz von Proxies: Aufhalten an der Firewall
  - „Gehärtete“ Anwendungen laufen lassen
- Schadensbegrenzung
  - Benutzerrechte minimieren
  - Möglichkeiten nach „Einbruch“ minimieren

# Härtung

- Härtung = Schutz vor allgemeinen Angriffen
- Die Programme die auf Firewall laufen sollen mit speziellen Massnahmen erstellen:
  - gcc mit „Stack Smashing Protection“ von IBM
  - reduzierte C-Bibliothek verwenden:  
„uClibc“  
(glibc ist inzwischen zum „Monster“ mutiert)
  - Sicherheitsbibliotheken einbinden:  
„libsafe“ von avayalabs  
andere
- Höherer Aufwand für einige wenige Programme

# Auswahl der System-Software(1)

- Was braucht ein Firewall Rechner überhaupt?
  - Bootloader: Boot von CD (initrd, read-only)
  - Kernel mit Netzwerkkarten-Treibern
  - ~~System / C - Bibliothek~~  
Programme statisch kompilieren
  - ~~Internet Zugang / Einwahl~~  
DSL-Router
  - ~~System-V init Prozedur~~  
Eigenes „init“ Programm
  - „ifconfig“ zur Netzwerk Konfiguration
  - „iptables“ zur Konfiguration des Paketfilters

# Auswahl der Anwendungs-Software (2)

- Server-Dienstprogramme:  
httpd, squid, fetchmail? Erstmal Nichts!
- Client Programme? Erstmal Nichts!
- KEIN ANDERES PROGRAMM  
auch keine
  - Konsole oder Shell
  - cp / rm / chmod / ...
  - Einfach nichts !!!
- Chroot und Einschränkung von  
Benutzerrechten entfällt

# Vorteile

- Minimalsystem gut kontrollierbar  
„wenig“ unkontrollierte Prozesse auf Rechner
- Härtung von Kernel und Programmen  
aufgrund geringer Quantität  
nicht besonders aufwendig
- Bildschirm / Festplatte / Floppy  
nicht notwendig

# Nachteile

- Zweistufige Erstellung
  - initrd -Image vorbereiten (und brennen)
  - Booten und testen (qemu)
- Kein Einloggen und Arbeiten auf Rechner möglich
- Aufgrund fehlender Festplatte minimal höherer Speicherbedarf

# Proof of Concept

- 1. April 2005 – Erster Erfolgreicher Einsatz  
Kein Aprilscherz:
  - Einsatz als Filtering Gateway zw. 2 Netzwerken
  - Derzeitig noch 109MB auf CD
    - Abspecken durch Entfernung von unnötigen Programmen ist geplant!

# LSFS - Details

- Verwendung von
  - ISOLINUX von SYSLINUX (Boot von CD)
  - Kernel 2.6.7 mit PaX Patch
  - BUILDROOT
    - eine Build-Umgebung für lauffähiges Minimalsystem
      - gcc 3.4.1 mit Stack Smashing Protector (SSP) Patch
      - uClibc (C-Bibliothek)
      - bash (für die Tests)
      - iptables
      - ifconfig (net-tools)
  - eigenem Init-Programm in C (kein Shellscript!)
    - Einhängen von /proc
    - Netzwerkinitialisierung (ifconfig / route)
    - Firewallkonfiguration (iptables)
    - bash (optional für Test der Firewall-Regeln)

# LSFS - Details

Was läuft nach dem Systemstart noch?

Nur unser Init-Programm in einer Endlosschleife:

```
while(1)
    pause();
```

und der Kernel.

# ToDo's

- 1) CD abspecken: entfernen nicht verwendeter Programme der BUILDROOT Umgebung.
- 2) Einbindung von Proxies (z.B. squid, (s)pop3 (s)smtp Proxy, rinetd)
- 3) Automatisierung von:
  - 1) Herunterladen der Pakete und Patches
  - 2) Patchen und Kompilierung von LSFS
  - 3) Abspecken des Testsystems

# Links

- BUILDROOT - <http://buildroot.uclibc.org>
- Stack Smashing Protector  
<http://www.tri.ibm.com/projects/security/ssp>
- ISOLINUX <http://syslinux.zytor.com>
- Libsafe  
<http://www.research.avayalabs.com/project/libsafe>
- PaX - <http://pax.grsecurity.net>
- Openwall Project - <http://www.openwall.com>
- Secure Programming HOWTO  
<http://www.dwheeler.com/secure-programs>
- Infocus - <http://securityfocus.com/infocus/1539>