Linux Security Basics

<christian.mock@vibe.at>

15.9.2001

Agenda

- 14:30 Linux Security Basics
- 18:00 PGP
- 19:30 Wieden Bräu

Pausen: aber sicher, ich brauch mein Nikotin

Fragen: nur keine Zurückhaltung

(es gibt nur blöde *Antworten*!)

/me

- seit 1992 Linux (0.98 oder so)
- 1994 ISP PING technisch aufgebaut
- 1997 Security−Abteilung bei Xsoft
  - Security Checks
  - Firewalls & Zubehör
  - Consulting
- VIBE!AT Gründungsmitglied

Einführung...

- Was ist Security?
  - “CIA”: Confidentiality, Integrity, Availability
  - Also nicht nur gegen böse Cracker, sondern auch gegen Schnüffler und Naturkatastrophen
- Warum brauch ich Security?
  - Scriptkiddies und Würmer suchen sich ihre Ziele nicht aus, sondern nehmen, was daherkommt
- Es gibt keine 100%ige Security!
  - Hauptaufgabe ist Risk Management

"security is a process" (Schneier?)

- risiko−analyse
- entscheidung über schutzmaßnahmen
- implementation derselben
- betrieb, up−to−date−halten
- überwachung
- back to square one

"principle of least privilege"

- wichtiges grundprinzip
- usern, prozessen, maschinen... nur soviel rechte geben wie notwendig
- z.B:
  - sparsam mit “other”−rechten im filesystem
  - firewalls als “default deny” konfigurieren
  - special−use−usern keine shell geben
  - daemonen nicht als root und chroot(2)ed

Wer ist der Gegner?

- interne vs externe
- cracker vs script kiddies
- würmer und andere automaten
- spammer
- beamtete und andere schnüffler
- der zufall

was ist eigentlich

- firewall
- intrusion detection
- https
- ssh
- VPN / IPSEC
- permissions, umask

firewalls

- übergangspunkt zwischen netzen verschiedenen trust−levels
- aller verkehr geht drüber −> einziger enforcement point, (relativ) gut unter kontrolle zu halten
- **hat keine wunderkräfte!**
  - wenn schlecht konfiguriert, hilfts nix
  - was durchgelassen wird, ist i.A. nicht gesichert

firewall−typen: application layer

- behandeln datenströme auf transport/application layer
- “plugs”
- können bei korrekter implementierung z.B. vor telnetd−buffer overflow schützen, aber kaum vor code red
- performen ned für high−volume−traffic (http) weil kernel−>userspace−>kernel
- Bsp: fwtk, plugger

firewall−typen: packet filter

- schauen IP−pakete an, mit seitenblick auf TCP/UDP header
- primitive PF sehen nur einzelne pakete
  - scheiße für UDP
  - TCP: checken ACK−bit, damit ACK−scans möglich
  - schnell wie hölle
- Bsp: Cisco ACL, ipchains/ipfwadm

firewall−typen: stateful packet filter

- versuchen, zusammenhang zwischen paketen zu sehen
- TCP: jenachdem, wie implementiert, besser oder schlechter (welcher SPF checkt sequence numbers?)
- UDP: heuristik “rückpaket muß innerhalb X sekunden kommen”
- recht schnell
- Bsp: iptables, ipfilter (*BSD), FireWall−1

firewall: auswahlkriterien

- auskennen
- preis
- GUI notwendig? ease of config
- logging−qualitäten
- nach bekannten security−holes recherchieren

intrusion detection

- wenn die firewall die verstärkte tür ist, ist das IDS die alarmanlage
- üblicherweise sniffer mit pattern−matching und mehr oder weniger buntem GUI
- sehr interessant, wenn man weiß, was das ding eigentlich meldet
- aufwand zum tunen nicht unerheblich
- warnung vor automatischen gegenschlägen
- Bsp: snort, RealSecure, CyberCop, ...

SSL

- crypto auf transport layer
- in applikation implementiert...
- ...oder gewrapped: sslwrap, stunnel
- verschlüsselung und authentisierung
- üblicherweise nur server gegenüber client authentisiert

SSL (2)

- für viele TCP−basierende protokolle heute unterstützt/definiert:
  - imap, pop3 (z.B. in netscape, fetchmail; server per wrapper)
  - nntp (netscape, inn)
  - ...
- recht einfach zum aufsetzen, wenn man openssl begriffen hat
- CERTs generieren nicht so trivial

https: http+SSL

- verbreitetste SSL−anwendung
- in fast allen browsern vorhanden
- daher sehr gut für web−apps zu verwenden
- üblicherweise authentisiert sich der server gegenüber dem client, um spoofing zu verhindern
- client kann sich auch authentisieren, damit strong auth
- apache: mod_ssl, apache−ssl

ssh

- “telnet on crypto−steroids”
- starke verschlüsselung
- starke authentisierung
- X11−forwarding
- port forwarding vorwärts und zurück
- ssh−agent: authenticate once, use many
- scp: sicheres file−kopieren

VPN

- “virtual private network”
- ersatz für standleitungen und private dialups
- arbeiten üblicherweise am network layer
- daher für applikationen transparent
- diverse ansätze: proprietäre, hacks (PPP over ssh), IPSEC...

IPSEC

- VPN−standard
- breite unterstützung
- meistens interoperabilität
- ziemlich komplex
- für linux: FreeS/WAN

permissions

- drwxrwxrwx
- file type (regular, dir, socket, char/block special...)
- user
  group
  other
  jeweils **r**ead **w**rite e**x**ecute

permissions auf files

- if(fileuser == user)
  return checkuserperms()
  else if(filegroup == group)
  return checkgroupperms()
  else
  return checkotherperms
- d.h. user mock darf file
  ----rwxrwx mock mock
  nicht lesen/schreiben/executen!

permissions auf dirs

- r: lesen, d.h. liste der files bekommen
- w: schreiben, d.h. files/subdirs anlegen/löschen
- x: chdir und generell befingern von subdirs
- d.h. permission zum löschen eines files hängt von den permissions des directories, in dem es ist, ab!

suid−bits auf files

- −r−sr−sr−x
- owner−suid: bei execute läuft prozess mit uid des file−owners
- group−suid: prozess läuft mit gid des group owners
- other−suid: AFAIK nicht mehr in verwendung

- suid−programme erlauben das ausführen unter anderen uids −> gefährlich!

suid−bits auf dirs

- owner−suid: noop, AFAIK
- group−suid:
  bei file−creation, wenn user die entsprechende secondary group hat, bekommt file diese group
- other−suid (“t”):
  directory−owner oder file−owner darf files löschen (/tmp: drwxrwxrwt)

umask

- welche bits der permission werden bei file−/dir−creation *nicht* gesetzt?
- “offener” standard: 022 (oktal), d.h. write für group/other wird nicht gesetzt
- “paranoid”: 077, d.h. keine permissions für group/other
- “dumm”: 0, d.h. −rwxrwxrwx

warum sollte ich nicht:

- dauernd als root arbeiten?
- telnet verwenden?
- rlogin/rsh verwenden?
- unverschlüsseltes POP3 verwenden?
- unverschlüsselte mails senden?
- mein OS in der "installiere alles"−Variante aufsetzen?

OS hardening

aka "ich hab mein *nix aufgesetzt, kann ich jetzt ans netz?"

warum?

- netzwerk−daemons
- standard−accounts
- software−config

wie finde ich raus, was überhaupt läuft?

- netstat −aA inet
  zeigt alle in−use inet sockets an, TCP LISTENING und udp sind interessant
- lsof −I *protocol@host:port*
  zeigt prozess dazu an
- portscan von anderem rechner aus
  - nmap
  - strobe
  - nessus

wie entscheide ich, ob ich das brauche?

- wir erinnern uns ans “priniciple of least priviledge”: wenn ich nicht sicher bin, daß ich’s brauch, weg damit!
- herausfinden, zu welchem paket der daemon gehört (“dpkg −S /file/name”) und doku lesen
- trial and error
- vorsicht bei abhängigkeiten, z.b. gnome und KDE

wie werd ich’s wieder los?

- am besten: paket deinstallieren, wenn nicht gebraucht
- herausfinden, woher gestartet:
  - /etc/rcX.d
  - /etc/inittab
  - sonstwo
- ...und deaktivieren

wie überprüf ich’s nachher?

- netstat
- portscan
- security scanner

nützliche Tools aufsetzen

- ssh (inkl key−mgmt)
- ipchains + port−forwarding
- apache + SSL
- logcheck
- snort
- tripwire

Betrieb (logs lesen, firewalls,...)

- Was gibt’s zu tun?
  - logs lesen
  - Unregelmaessigkeiten investigieren
  - tripwire laufen lassen
  - software up−to−date halten:
    - bugtraq@securityfocus.com,
    - vendor−security−mailinglists,
    - debian "apt−get update/upgrade"