Sichere Systeme mit OpenBSD

Alexander von Gernler

<grunk@openbsd.org>

5. Erlanger Linuxtage, 24./25. März 2007

Inhaltsübersicht

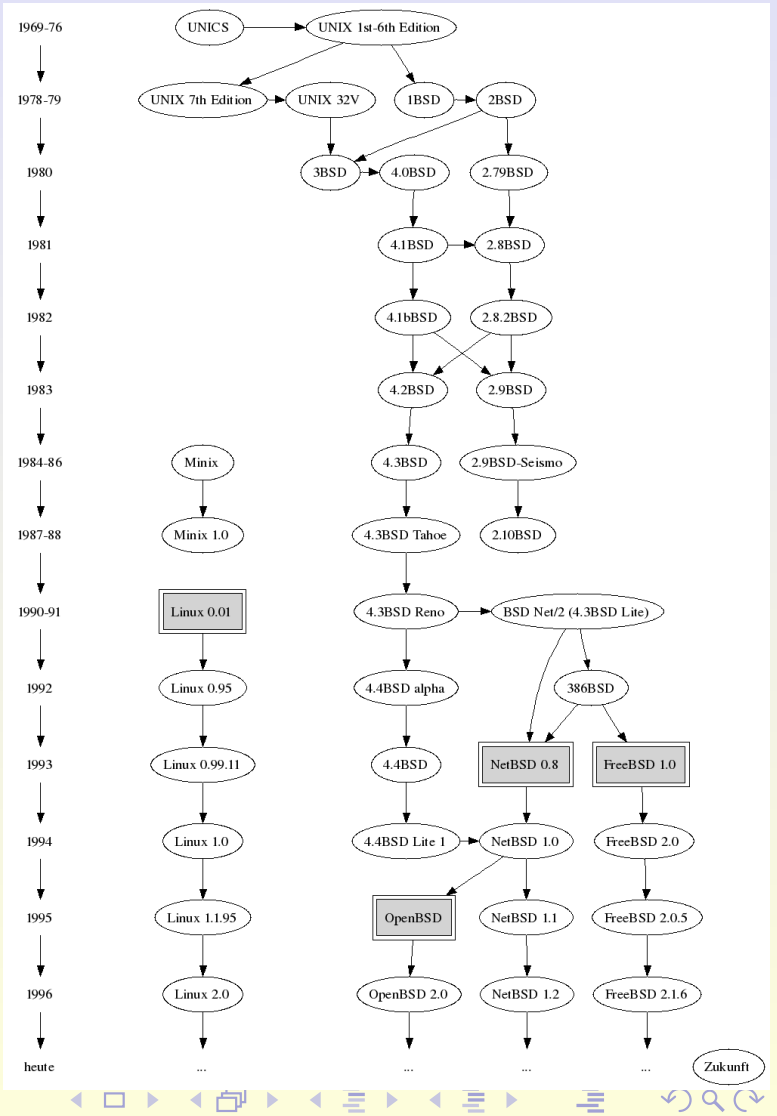
Inhaltsübersicht

Über den Vortrag

- **Dieser Vortrag ist...**
  - eine Vorstellung von OpenBSD im Unterschied zu Linux
  - eine Diskussion der Exploit-Schutzmechanismen von OpenBSD
  - ein subjektiver Erfahrungsbericht
- **Er ist nicht...**
  - ein Installations-HOWTO für OpenBSD
  - eine technische Referenz
  - ein heiliger Kreuzzug
- **Der Autor...**
  - ist OpenBSD-Entwickler
  - benutzt OpenBSD seit Version 2.9
  - betreibt den OpenBSD Mirror an der Uni Erlangen

Was ist eigentlich BSD?

- **BSD:** Nachfolger des Ur-Unix
- **Linux:** Nachempfunden von Torvalds, begonnen während seines Studiums
- Beide implementieren POSIX-Standards und sind damit unixoide Betriebssysteme.
- Quellcode, der auf Linux baut, baut meist auch unter BSD
- Softwareauswahl steht wie gewohnt zur Verfügung
  - Mozilla, MPlayer, XMMS, KDE, Apache, MySQL, Gimp, ...



Wieviele Distributionen gibt es denn?

- Es gibt keine Distributionen!
- Drei große Projekte (sog. Flavors)
  - FreeBSD
  - NetBSD
  - OpenBSD
- Jedes Projekt stellt exakt eine Distribution bereit
- Trennung von Kernel und Userland wenig sinnvoll, trotz Versuchen wie Debian mit BSD-Kern o. ä.
- Kein Distributionen-Wildwuchs wie bei Linux (dreistellige Zahl, so genau weiss das niemand)

Entwicklungsmodus

- Quellcodebaum per CVS versionsverwaltet
- Kommunikation über internen Chat und interne Mailingliste
- Kein Commit ohne OK von anderen Entwicklern (wenige Ausnahmen)
- Letztes Wort hat Theo de Raadt
- Fehlverhalten wird mit Accountsperrung belohnt
- Demokratie ist prinzipiell toll, funktioniert aber nicht bei Softwareprojekten: Machen statt reden
- Benutzeranfragen werden meist mit “Submit a diff” beantwortet

Philosophie von OpenBSD

- Absolut freie Lizenz erlaubt Verwendung des Quellcodes für *jeden* Zweck
- Oberste Priorität auf Sicherheit
- Integration starker Kryptographie mit der Möglichkeit, sie überall hin zu exportieren
- Die beste Entwicklungsplattform überhaupt bereitstellen
- Standards verfolgen und korrekt implementieren (POSIX, ANSI, X/OPEN etc.)
- Zielgruppe: Eigene Entwickler
- Absolut konservative Entwicklungsstrategie: Features lieber nicht als schlecht implementieren

Erklärte Nicht-Ziele von OpenBSD

OpenBSD will nicht...

- möglichst viele Benutzer haben
- einen grafischen Installer schreiben
- seinen “Marktanteil” vergrößern
- immer den letzten Hype unterstützen
- XML/PHP/Java/Web 2.0 Webseiten haben
- Logo-Contests ausrufen
- Fragwürdigen Code in den Tree nehmen, nur um Preise zu gewinnen
- Binärtreiber aufnehmen, nur um Nutzer zufriedenzustellen
- die Lösung für alles sein

Lizenzdiskussion: GNU GPL vs. BSD-style

- **GNU GPL**
  - Will größtmögliche Freiheit für die Community
  - Weitergabe der Software nur mit Weitergabe des Quellcodes, dies muss ebenfalls unter GPL erfolgen
  - Hält die Quellen rechtlich gesehen für immer offen
  - Aber: Prozess gegen Router-Hersteller zeigt, dass GPL auch von Firmen absichtlich missachtet wird
- **BSD-style**
  - Will größtmögliche Freiheit für den Einzelnen
  - Weitergabe der Software unter jeder beliebigen Lizenz, auch Closed Source möglich
  - Wichtig nur: Wahrung des Copyrights
  - Große Verbreitung von BSD Code bis zu Windows (TCP Stack)

Inhaltsübersicht

Technische Daten

- Freies Open Source UNIX, basierend auf den 4.4BSD Quellen
- Eigenes Binärformat, aber Emulation für Linux, FreeBSD, Solaris (SVR4), BSD/OS, SunOS und HP/UX
- Plattformen: i386, amd64, macppc, sparc, sparc64, zaurus, alpha, landisk, vax, mac68k, sgi, hp300, hppa, mvme68k, mvme88k, luna88k, armish

- Standardmäßig IPv6- und IPsec-fähig
- Minimalinstallation ca. 100 MB, Standardinstallation bei ca. 300 MB, typisches Desktop-System bei ca. 2 GB
- Software installierbar als Binärpakete oder aus Ports Collection. Derzeit über 4209 Ports

Wo kann man OpenBSD einsetzen?

- Sicherheitskritische Anwendungen
  - Internet Service Provider
  - Firmennetzwerke
- Firewalls
  - Selbst aufgesetzte Firewall
  - auf OpenBSD basierende Produkte
- Sichere Workstations
  - Installation ist *secure by default*
- Sicherheitsbewusste Softwareentwicklung
  - Paranoide Laufzeitumgebung fuer Binaries
  - hochqualitative Manpages
  - aufpolierter Quellcode-Checker `lint(1)`

OpenBSD beziehen

- Auslieferung auf 3 CDs jedes halbe Jahr (feste Releasezyklen im Mai und November)
  - Schickes Artwork mit Aufklebern
  - Neuer OpenBSD Release Song
  - Kostenpunkt ca. EUR 50,-
  - Erhältlich von `openbsd.org` oder über Buchhandel
- Per Diskette, CD oder PXE bootbar, Installation der Pakete aus dem Netz kostenlos

- Großes Netz an tagesaktuellen Mirrors weltweit verfügbar
- Bereitstellung von Security-Patches und Bekanntgabe auf `security-announce@openbsd.org`

Wo kommt die tolle Artwork immer her?

- Ty Semaka, Mediendesigner und Bekannter von Theo
- Bandleader der “Plaid Tongued Devils”
- Zeichnet Puffy und seine Freunde
- Komponiert und spielt die OpenBSD Songs ein
- Jetzt zur OpenBSD Version 4.0 erschienen: Compilation aller OpenBSD Songs von 3.0 bis 4.0, und ein Interview mit Ty

Technische Gemeinsamkeiten

"BSD = Linux with a twist" (Christian Weisgerber)

- Oberflächlich: Einschalten, Bootloader, Kernel, Init, Textkonsole mit Login, X11 möglich
- Alle Standardbefehle vorhanden (`ps(1)`, `top(1)`, `ifconfig(8)`, `ping(8)`, `vi(1)`, `...`)
- Wichtige Internet-Daemons im Basissystem: `httpd(8)`, `sshd(8)`, `ftpd(8)`, `named(8)`, `sendmail(8)`, `ntpd(8)`, `identd(8)`, `nfsd(8)`, `inetd(8)`
- Breite Hardwareunterstützung, auch Cryptohardware (Beschleuniger, z. B. hifn)
- Gängige Software als Ports vorhanden: MySQL, PostgreSQL, Gimp, xmms, MPlayer, Mozilla, KDE, GnuPG, mutt

Technische Unterschiede

- BSD Fast Filesystem `ffs` statt Linux `ext2/ext3`
- Erweiterung *SoftUpdates* sorgt für Geschwindigkeit und Konsistenz bei Absturz, ist aber nicht mit Journaling (vgl. `ext3`) gleichzusetzen
- Die meisten Linux Distros haben ein System V `init` mit Runlevels, die BSDs haben das einfachere BSD-style `init`
- Kernel und Userland bilden feste Einheit und können nur gemeinsam aktualisiert werden. Sie passen deshalb immer zusammen. Keine Probleme wie:
  - `lvmtools` passen nicht zum Kernel LVM
  - `iptables` will andere Netfilter Version im Kern
  - `reiserfsck` hat mit dem alten Kern doch noch getan?!

Andere Kernel-Philosophie

- Kernelmodule werden praktisch nicht benutzt
  - Kein einschleusen von Code ins laufende System
  - Probleme, immer passendes Modul zu laden (muss ohnehin auf gleicher Codebasis kompiliert werden) entfallen
  - Stabiler, sicherer
- Kernel customizen ist No-No: `GENERIC` erfüllt alle Wünsche
  - Kein Gewinn durch Weglassen von Komponenten (weder Geschwindigkeit, noch Platz) auf heutigen Maschinen
  - Keine Fehler, die nur durch falsch konfigurierte Kernel entstehen
  - Entwickler leisten nur Support für `GENERIC`
- Compilerflags und andere Geschwindigkeitshacks werden extrem abgelehnt
- Aufgaben von `/proc` und `/kern` werden von `sysctl(1)` übernommen

Was ist neu in OpenBSD 4.1?

- Neue Plattform `landisk`
- Verbesserte Unterstützung für Suns UltraSparc III
- `spamd(8)` kann jetzt Greylisting über mehrere Hosts
- Unterstützung von noch mehr USB-Massenspeichergeräten

- Support für diverse Funkuhren und GPS-Geräte
- `ntpd(8)` kann solche Zeitquellen nutzen
- `hoststated(8)` macht Layer 3 und Layer 7 Loadbalancing
- OpenSSH 4.6

Offene Rechnungen: Blob

- Praktisch alle anderen Distributionen (auch BSDs) benutzen Binärtreiber (NVidia, Atheros, Broadcom, Adaptec, ICP Vortex, 3-Ware, Intel, IBM, ...)
- Binärtreiber widersprechen **komplett** dem OpenSource Gedanken
- Distributionen machen es trotzdem, weil es *bequem* für die Benutzer ist
- SHAME ON YOU!

Offene Rechnungen: Blob

- Praktisch alle anderen Distributionen (auch BSDs) benutzen Binärtreiber (NVidia, Atheros, Broadcom, Adaptec, ICP Vortex, 3-Ware, Intel, IBM, ...)
- Binärtreiber widersprechen **komplett** dem OpenSource Gedanken
- Distributionen machen es trotzdem, weil es *bequem* für die Benutzer ist
- **SHAME ON YOU!**

Offene Rechnungen (2): Free Linux Driver Development!

- Mail von Greg Kroah-Hartman an die Linux Kernel Mailingliste
  `http://article.gmane.org/gmane.linux.kernel/487536`
- NDAs sind für die Linux-Entwickler auch kein Problem:

> ‘‘If your company is worried about NDA issues surrounding your device’s specifications, we have arranged [...] to properly assure that all needed NDA requirements are fulfilled.’’

- Kommt eine Firma bei Linux mit NDA durch, gibt es auch für die anderen keine freie Doku mehr
- Komplett gegen den *Geist* der GPL!
- Siehe Quellcode des „freien“ X.org `nv`-Treibers:
  `xc/programs/Xserver/hw/xfree86/drivers/nv/nv_setup.c`
- **SHAME ON YOU!**

Offene Rechnungen (2): Free Linux Driver Development!

- Mail von Greg Kroah-Hartman an die Linux Kernel Mailingliste
  `http://article.gmane.org/gmane.linux.kernel/487536`
- NDAs sind für die Linux-Entwickler auch kein Problem:

> ‘‘If your company is worried about NDA issues surrounding your device’s specifications, we have arranged [...] to properly assure that all needed NDA requirements are fulfilled.’’

- Kommt eine Firma bei Linux mit NDA durch, gibt es auch für die anderen keine freie Doku mehr
- Komplett gegen den *Geist* der GPL!
- Siehe Quellcode des „freien“ X.org `nv`-Treibers:
  `xc/programs/Xserver/hw/xfree86/drivers/nv/nv_setup.c`
- **SHAME ON YOU!**

Offene Rechnungen (2): Free Linux Driver Development!

- Mail von Greg Kroah-Hartman an die Linux Kernel Mailingliste
  `http://article.gmane.org/gmane.linux.kernel/487536`
- NDAs sind für die Linux-Entwickler auch kein Problem:

> ‘‘If your company is worried about NDA issues surrounding your device’s specifications, we have arranged [...] to properly assure that all needed NDA requirements are fulfilled.’’

- Kommt eine Firma bei Linux mit NDA durch, gibt es auch für die anderen keine freie Doku mehr
- Komplett gegen den *Geist* der GPL!
- Siehe Quellcode des „freien“ X.org nv-Treibers:
  `xc/programs/Xserver/hw/xfree86/drivers/nv/nv_setup.c`
- **SHAME ON YOU!**

Offene Rechnungen (2): Free Linux Driver Development!

- Mail von Greg Kroah-Hartman an die Linux Kernel Mailingliste
  `http://article.gmane.org/gmane.linux.kernel/487536`
- NDAs sind für die Linux-Entwickler auch kein Problem:

> ‘‘If your company is worried about NDA issues surrounding your device’s specifications, we have arranged [...] to properly assure that all needed NDA requirements are fulfilled.’’

- Kommt eine Firma bei Linux mit NDA durch, gibt es auch für die anderen keine freie Doku mehr
- Komplett gegen den *Geist* der GPL!
- Siehe Quellcode des „freien“ X.org `nv`-Treibers:
  `xc/programs/Xserver/hw/xfree86/drivers/nv/nv_setup.c`
- SHAME ON YOU!

Offene Rechnungen (2): Free Linux Driver Development!

- Mail von Greg Kroah-Hartman an die Linux Kernel Mailingliste
  `http://article.gmane.org/gmane.linux.kernel/487536`
- NDAs sind für die Linux-Entwickler auch kein Problem:

> ‘‘If your company is worried about NDA issues surrounding your device’s specifications, we have arranged [...] to properly assure that all needed NDA requirements are fulfilled.’’

- Kommt eine Firma bei Linux mit NDA durch, gibt es auch für die anderen keine freie Doku mehr
- Komplett gegen den *Geist* der GPL!
- Siehe Quellcode des „freien“ X.org `nv`-Treibers:
  `xc/programs/Xserver/hw/xfree86/drivers/nv/nv_setup.c`
- **SHAME ON YOU!**

Was möchten wir erreichen?

- Alle Exploits bekannter Software nützen unvorhergesehene Zustände aus
- Solche Zustände entstehen durch Fehler in der Software
- Weniger Bugs bedeuten also weniger Exploits
- Es wäre toll, diese *nebenher* finden zu können
  - OpenBSD hilft hier, vor allem bei C/C++
  - Perl, Python und Co.: OpenBSD findet Fehler im *Interpreter*!

Wie hilft uns das Betriebssystem?

- Toolchain
  - Erweiterungen im GCC, siehe `gcc-local(1)`
    - `-Wbounded`
    - `-Wstack-larger-than-N`
    - ProPolice StackSmashing Protection
  - verbesserter Quellcodechecker `lint(1)`
- Ablaufumgebung
  - Write XOR Execute
  - random `mmap()`/`malloc()`
  - random library allocation
  - Stackgap
  - non-executable Stack

Oh mein Gott, es lebt!

- Die meisten Programme unter Linux laufen eher zufällig

- `malloc()` nach `malloc()` liefert zusammenhängenden Speicher?
- Das erste `malloc()` in einem Programm liegt immer an der selben Adresse?
- Mit `malloc()` geholter Speicher ist immer mit `0` initialisiert?
- Mit `free()` wieder freigegebener Speicher kann noch geschrieben werden?
- Zugriff über Arraygrenzen hinweg (off-by-one) ist kein Problem?

Oh mein Gott, es lebt!

- Die meisten Programme unter Linux laufen eher zufällig

- `malloc()` nach `malloc()` liefert zusammenhängenden Speicher?
- Das erste `malloc()` in einem Programm liegt immer an der selben Adresse?
- Mit `malloc()` geholter Speicher ist immer mit `0` initialisiert?
- Mit `free()` wieder freigegebener Speicher kann noch geschrieben werden?
- Zugriff über Arraygrenzen hinweg (off-by-one) ist kein Problem?

Oh mein Gott, es lebt!

- Die meisten Programme unter Linux laufen eher zufällig

- `malloc()` nach `malloc()` liefert zusammenhängenden Speicher?
- Das erste `malloc()` in einem Programm liegt immer an der selben Adresse?
- Mit `malloc()` geholter Speicher ist immer mit `0` initialisiert?
- Mit `free()` wieder freigegebener Speicher kann noch geschrieben werden?
- Zugriff über Arraygrenzen hinweg (off-by-one) ist kein Problem?

Oh mein Gott, es lebt!

- Die meisten Programme unter Linux laufen eher zufällig

- `malloc()` nach `malloc()` liefert zusammenhängenden Speicher?
- Das erste `malloc()` in einem Programm liegt immer an der selben Adresse?
- Mit `malloc()` geholter Speicher ist immer mit `0` initialisiert?
- Mit `free()` wieder freigegebener Speicher kann noch geschrieben werden?
- Zugriff über Arraygrenzen hinweg (off-by-one) ist kein Problem?

Oh mein Gott, es lebt!

- Die meisten Programme unter Linux laufen eher zufällig

- `malloc()` nach `malloc()` liefert zusammenhängenden Speicher?
- Das erste `malloc()` in einem Programm liegt immer an der selben Adresse?
- Mit `malloc()` geholter Speicher ist immer mit `0` initialisiert?
- Mit `free()` wieder freigegebener Speicher kann noch geschrieben werden?
- Zugriff über Arraygrenzen hinweg (off-by-one) ist kein Problem?

Wenn es kompiliert, liefern wir es aus!

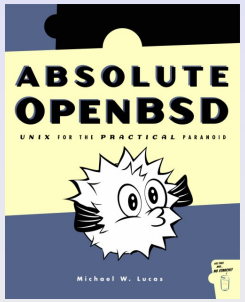
- Viele Programme laufen unter Linux, obwohl sie Fehler haben
- OpenBSD lässt solche Programme von vornherein brechen
  - Mozilla-Port nach OpenBSD dauerte ewig
  - OpenOffice läuft jetzt endlich, solala
  - Auch heute noch hat viel Freie Software kaputte Signal-Handler
  - Programme starten und sofort mit `^C` abbrechen, schiesst
    - xpdf
    - xmms
    - ...
- Dadurch Chance die Fehler zu entdecken und zu beheben
- Softwareentwicklung unter OpenBSD macht Programme sicherer

Läuft's auch unter sparc64?

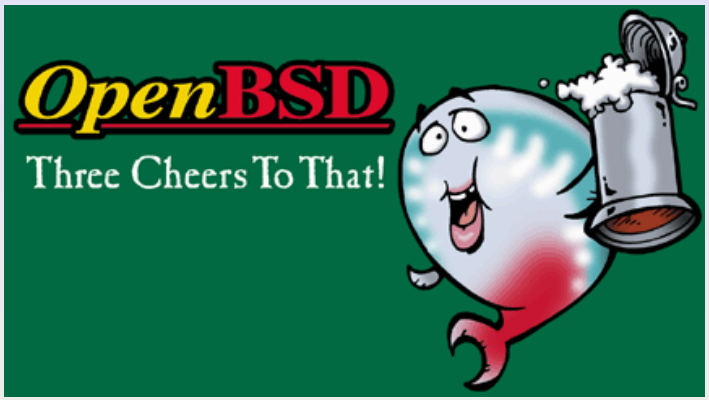
- Das Testen auf mehreren Architekturen findet noch mehr Fehler
- Gut selbst dann, wenn das eigene Produkt nur auf einer Plattform laufen soll
- Viele Hindernisse fallen plötzlich ins Auge:
  - 32bit vs. 64bit
  - Pointer Size, Page Size
  - Little Endian vs. Big Endian
  - Alignment
  - Virtuelle Adressen für DMA
  - Stack-Wachstum nach oben oder unten?
- Idealer Mix: `i386`, `amd64`, `macppc`, `sparc64`

Bücher zu OpenBSD

- **Absolute OpenBSD**
  - von Michael Lucas
  - gutes Kompendium, sehr humorvoll
  - ISBN 1-886411-99-9
- **Secure Architectures with OpenBSD**
  - von Brandon Palmer und Jose Nazario
  - kein stringentes Werk, aber gutes Kochbuch
  - ISBN 03-21193-66-0
- **Building Firewalls with OpenBSD and PF [2nd edition]**
  - von Jacek Artymiak
  - deckt ein Spezialthema ab
  - ISBN 83-916651-1-9



Noch Fragen?



1. Folien erstellt mit LaTeX, `latex-beamer`, `make` und CVS unter OpenBSD/i386
2. Folien erhältlich unter `http://pestilenz.org/~grunk/vortraege/2007/03-erlug/openbsd.pdf`
3. Quellcode der Folien auf Anfrage: ⟨`grunk@pestilenz.org`⟩

`$Id: openbsd.tex,v 1.22 2007/03/25 09:23:31 grunk Exp $`