



Angriff

Abhärtung von Linux-/BSD-Systemen – eine Einführung

Steffen Wendzel

Schwierigkeitsgrad



Dieser Artikel stellt eine Anleitung für die schnelle Absicherung von Linux-/BSD-Systemen dar. Der Inhalt kann ohne viel Vorkenntnisse auf dem eigenen System angewandt werden.

Ein Linux-/BSD-System – zumindest grundlegend – abzuhärten ist gar nicht so schwer. Dieser Artikel stellt im Wesentlichen ein Tutorial dar um die Abhärtung auf dem eigenen Rechner zu bewerkstelligen.

Bevor es los geht

Bevor mit der eigentlichen Abhärtung (engl. *hardening*) begonnen wird sollte dafür gesorgt werden, dass das zu härtende System sich in einem nicht kompromittierten Zustand befindet. Das wird am einfachsten durch eine Neuinstallation bewerkstelligt.

Nach der Auswahl sollten alle installierten Packages des Systems updated werden, so dass keine bereits bekannten Sicherheitlöcher vorhanden sind. Es ist zudem ratsam keine Beta-Versionen oder Unstable-Releases zu verwenden, sondern eine Stable-Release einer Distribution zu verwenden und in Zukunft nur Security-Updates zu installieren.

Auswahl der Distribution / des Derivates

Abhärten lässt sich generell jede Linux Distribution. Bei einigen ist der Aufwand allerdings

höher als bei anderen. Die Standarddistributionen verfügen über eine vergleichsweise geringe Absicherung im Vergleich zu speziellen Sicherheitsdistributionen, deren Pakete und Kernel in der Regel explizit gehärtet sind. Zudem sind bei diesen Distributionen restriktivere Zugriffsrechte im Dateisystem zu finden. Doch auch die Standarddistributionen sind im Laufe der letzten Jahre deutlich sicherer geworden.

Die meisten Security-Distributionen basierend auf sehr bekannten Mainstream-Di-

In diesem Artikel erfahren Sie...

- XXXXXXXXXXXXXXXXXXXX;
- XXXXXXXXXXXXXXXXXXXX;
- XXXXXXXXXXXXXXXXXXXXn.

Was Sie vorher wissen/können sollten...

- Linux Grundlagen ;
- Grundlegendes Verständnis für die Sicherheitskonzepte von Linux.

istributionen, was sehr sinnvoll ist, da die zahlreichen Benutzer solcher Mainstream-Distributionen mit dem vorhandenen KnowHow auch die Security-Distribution bedienen können.

Für Benutzer von Debian gibt es beispielsweise Adamantix. Ebenfalls recht populär ist Hardened Gentoo, das OpenWall Projekt und Immunix. Ich selber leite das *Hardened Linux* Projekt – eine Security-Distribution, die auf Slackware basiert.

Wer hingegen ein sicheres BSD-Derivat bevorzugt, der sollte sich OpenBSD ansehen.

Absicherung des Dateisystems

Die Absicherung des Dateisystems ist ein sehr wichtiger Schritt. Es geht hierbei zunächst darum Zugriffsrechte so anzupassen, dass die Möglichkeiten eines Angriffs minimal werden.

Das Programm 'find' ist für solche Aufgaben eine große Hilfe. Mit ihm kann beispielsweise nach Dateien gesucht werden, die für alle Benutzer des Systems schreibbar sind. Generell sollte man solche Zugriffsrechte (ausser evtl. in */tmp*) nicht zulassen.

Auf einem Bastion-Host können zudem die meisten (wenn nicht gar alle) SUID- und GUID-Bits von Programmdateien entfernt werden. Entfernt man via *chmod -s /bin/ping* beispielsweise das SUID-Bit von ping, so ist es einem Benutzer nicht mehr möglich das Programm zu benutzen, da ohne Root-Rechte keine ICMP Echo-Request Pakete versendet werden können.

Der nächste Schritt besteht darin überflüssige Binaries zu löschen.

Listing 1. Überprüfung auf vorhandene Stack Smashing Protection im gcc

```
$ echo | cc -fstack-protector-
                                all \
    -dM -E -|grep SSP
#define __SSP_ALL__ 2
```

Jede Binary, und jedes Paket, das eigentlich nicht benötigt wird gehört nicht auf einen Bastion-Host. Die Standarddistributionen installieren generell deutlich mehr Packages, als man benötigt. Das liegt allein schon darin, dass die meisten Pakete mit allen möglichen Abhängigkeiten übersetzt werden.

Ebenfalls sehr sinnvoll ist es, nach Dateien zu suchen, die Benutzern gehören, die nicht (mehr) existieren. Das ist sehr problematisch, da Benutzer, die man neu anlegt zufällig die gleiche Benutzer-ID bekommen können, wie die UID, die der Datei zugeordnet wurde. Die Folge dessen ist, dass ein Benutzer mit dieser Datei alles anstellen kann. Solche Dateien findet man am einfachsten mit einem *find / -path /proc -prune -o -nouser -o -nogroup*.

Weitere Möglichkeiten zur Härtung des Dateisystems stellen noch Access Control Lists (Erweiterte Zugriffsrechte), die Verschlüsselung von einzelnen Dateien (etwa mit gnupg) oder ganzen Dateisystemen (bspw. mit loop-AES, dm-crypt oder ab Kernel 2.6.19.0 eCryptFs) dar. Auch der SWAP lässt sich verschlüsseln, sofern man *mount* die Option *encrypted* übergibt.

Zudem können die Optionen beim Mounten von Dateisystemen verwendet werden, die zusätzliche Sicherheit bieten. Dazu zählen bspw. das Read-Only Mounting, das Schreibzugriffe verweigert. Ebenfalls nützlich ist *noexec*, das ausführbare Dateien verhindert (was für */home*-Partitionen sehr sinnvoll ist). *nosuid* verhindert die Benutzung von SUID-Bits, *nodev* verhindert die Erstellung von Gerätedateien.

Möchte man noch einen Schritt weiter gehen, so kann man die Veränderungen des Dateisystems auch überwachen. In meinem letzten Artikel zum Thema (Host Intrusion Detection) (Hakin9 2/07) habe ich bereits das Dateisystem-IDS (AIDE) vorgestellt, mit dem eine vernünftige Überwachung des Dateisystems möglich ist.

Schutz vor Speicherangriffen

Wahrscheinlich kennt jeder Leser so genannte Buffer-Overflow-Angriffe bei denen gezielt Speicherbereiche so überschrieben werden, dass introjizierter Code ausgeführt wird. Ebenfalls sehr bekannt sind Return-to-Libc-Angriffe, bei denen vom Angreifer versucht wird, gezielt bestimmte Library-Funktionen aufzurufen. Es gibt noch einige weitere solcher Angriffe auf Speicherbereiche, doch diese sollen an dieser Stelle nicht erwähnt werden. Vielmehr soll erklärt werden, wie man dieser Gefahr möglichst einfach aus dem Weg geht.

Zu diesem Zweck werde ich zwei Modifikationen vorstellen. Die Eine ist die ProPolice für den GNU C Compiler, die Andere ist ein Schutz für den Linux-Kernel und wird PaX genannt. OpenBSD, wie auch die meisten Security-Distributionen, bringt solcherlei Features im Übrigen schon von Haus aus mit. OpenBSD beinhaltet dabei neben gcc-ProPolice noch *W^X* (um Speicherseiten entweder schreib- oder ausführbar zu machen) und eine ganze Reihe anderer nützlicher Modifikationen, die auf <http://www.openbsd.org/security.html> gelistet sind.

Zunächst widmen wir uns der Compilermodifikation. Diese braucht nur vorgenommen werden, falls man selber Anwendungen übersetzen möchte und bereits die mit dem gcc erzeugten Programmdateien auf Stack Smashing Protection auslegen möchte.

Der ProPolice Patch muss in den alten Versionen des gcc noch von Hand eingepatcht werden. In neueren Versionen ist er bereits enthalten. Ob Ihr gcc über diese Fähigkeit verfügt erfahren Sie, indem Sie einen kleinen Testaufruf starten. Das folgende Listing zeigt, wie dies bewerkstelligt wird. Sollte die Preprozessoranweisung `#define` nicht erscheinen, so muss eine entsprechende Modifikation am gcc vorgenommen (die Anleitung ist hier zu finden: <http://www.trl.ibm.com/>



`projects/security/ssp/`) oder eine neuere Version installiert werden.

Um nun ein Programm so zu kompilieren, damit es eine Stack Smashing Protection bekommt und später auch von PaX geschützt werden kann, benutzen wir die Parameter `-fstack-protector-all` (zur Aktivierung des Stack-Schutzes) und `-pie` (zur Erzeugung von positionsunabhängigem Code in ausführbaren Dateien).

Ein Testprogramm soll nun demonstrieren und verifizieren, dass der Schutz aktiv ist.

Realisiert wird der ProPolice-Schutz indem zum Einen die lokalen Funktionsvariablen auf dem Stack vor den Buffer-Variablen plaziert werden, so dass ein Überschreiben der Buffervariablen nicht auch noch diese überschreibt. Zum Anderen wird hinter den Buffervariablen ein Canary-Wert integriert,

Listing 2.

```
XXXXXXXXXXXXXXXXXXXXXXXXX
$ cat test.c
#include <stdio.h>
#include <string.h>

int
main(int argc, char *argv[])
{
    char a[10];

    strcpy(a, argv[1]);

    return 0;
}
$ gcc -o test test.c
$ file test
test: ELF 32-bit LSB executable,
Intel 80386, version 1 (SYSV),
dynamically linked (uses shared
libs), not stripped
$ ./test `perl -e 'print
                                "#x100;'"`
Segmentation fault
$ gcc -o test -fstack-protector\
-pie test.c
$ file test
test: ELF 32-bit LSB shared
object, Intel 80386, version 1
(SYSV), not stripped
$ ./test `perl -e \
'print "#x100;'"`
*** stack smashing detected ***:
test terminated
Illegal instruction
```

der via `/dev/random` einen Zufallswert bekommt. Wird der Buffer überschrieben um bspw. den Wert des Befehlsregisters (EIP) zu überschreiben – auf diese Weise werden Buffer-Overflows ausgenutzt –, so wird auch der Canary-Wert überschrieben. Dadurch, dass der Canary-Wert jedoch auf eine Veränderung hin überprüft wird, kann mit sehr hoher Wahrscheinlichkeit festgestellt werden, ob ein Buffer-Overflow statt fand.

Doch nun zum kernelseitigen Schutz. PaX ist Bestandteil des GRSecurity-Patches (grsecurity.net) und stellt verschiedene Schutzfunktionen des Stacks zur Verfügung. Eine Einführung findet sich auf pax.grsecurity.net.

Zur Installation von GRSecurity lädt man zunächst einen aktuellen Kernel-Code sowie eine passenden Kernel-Patch von GRSecurity.net herunter.

Nachdem der Kernel (hier Version 2.6.18.6 von Hardened-Linux) entpackt wurde wird der Patch mittels `patch -p1 -i ../grsecurity-2.1.9-2.6.18.2-200611100917.patch` eingespielt. Anschließend wird der neue Kernel via `make config && make && make modules_install` installiert. Eventuell ist noch der Bootloader anzupassen und der bisher verwendete Kernel zu sichern sowie eine Notfall-Bootoption für den alten Kernel zu erhalten. Statt der blanken Text-Konfiguration des Kernels via 'make config' kann je nachdem, mit welchen Libraries und welcher Software das System ausgestattet ist auch `make menuconfig` für die ncurses-Variante oder auch 'make xconfig' für die X11-Variante verwendet werden.

Die Features von PaX und GRSecurity findet sich in der Kernelkonfiguration im Bereich (Security). Für PaX muss entweder die Option `PAGEXEC` oder `SEGME-XEC` aktiviert werden, und zudem

GRSecurity

Wenn nun sowieso der GRSecurity-Patch in den Kernel kompiliert werden soll, kann man sich auch gleich

noch dessen zahlreiche Features ansehen. Dazu zählt beispielsweise ein Role-Based Access Control System (RBACS), dass Restriktionen für die Programme des Systems erheben kann und mit dem Tool GRSecurity-Tool gradm konfiguriert werden kann. Solche Restriktionen empfehlen sich jedoch nur im Extremfall, weshalb ich in diesem Artikel weder auf GRSecurity-RSAC, noch auf SeLinux, AppAmor oder ähnlich Features eingehen werde.

Weitere Features von GRSecurity (die zum Großteil auch schon Solar Designers OpenWall Distribution enthalten waren) sind unter Anderem die Härtung von `chroot()`-Umgebungen, Auditing, besserer Schutz des `/tmp-Verzeichnisses` und auch ein Feature, dass es Benutzern nur erlaubt, ihre eigenen Prozesse zu sehen.

TCP/IP Stack absichern

Ein weiterer besonders wichtiger Schritt zum sichereren Linux-/BSD-System ist die Absicherung des TCP/IP-Stacks. Dies kann entweder via `sysctl` (Linux und BSD) oder via `/proc`-Dateisystem erledigt werden. Im Verzeichnis `/proc/sys/net/ipv4/conf` bzw. `/proc/sys/net/ipv6/conf` finden sich verschiedene Dateien in die ganz einfach via `echo n > Dateiname` ein Wert `n` (in der Regel eine `1` für die Aktivierung eines Features oder eine `0` für dessen Deaktivierung) eingetragen wird.

Die Variable `accept_redirects` sollte, sofern nicht grad ICMP Redirects akzeptiert werden immer auf `0` gesetzt werden: `echo 0 > /proc/sys/net/ipv{4,6}/conf/accept_redirects`. Das gleiche gilt für Ipv6 Router Advertisements (`accept_ra`).

ICMP Echo-Broadcasts können mit `icmp_echo_ignore_broadcasts` deaktiviert werden. Das Weiterleiten von Paketen und das Senden von Redirects wird mit `ip_forward` bzw. `send_redirects` kontrolliert. Es gibt noch einige weitere Variablen (die meisten unter ähnlichem Namen übrigens auch in *BSD, Solaris und ähnlichen Systemen), die das

Verhalten des TCP/IP-Stacks noch weiter beeinflussen können.

TCP-/UDP-Dienste liefert netstat -na Auskunft.

Deaktivierung von Netzwerkdiensten

Auf den meisten Systemen laufen standardmäßig nicht benötigte Dienste, etwa ein echo-Dienst via inetd, ein remote zugänglicher Sendmail-Dienst, oder auch RPC-Dienste. Diese sind auf Security-Distributionen wie Hardened Linux in der Regel standardmäßig deaktiviert. Ist dem nicht so, so sollte man sich die Mühe machen, und die nicht benötigten Dienste entweder deaktivieren oder komplett deinstallieren. Die in inetd aktivierten Dienste bekommt man ganz einfach via `egrep -v '^#.*$|^$' /etc/inetd.conf` heraus. Abstellen lassen sich die Dienste durch Auskommentierung mittels Raute und anschließendem Neustart des inetd: `kill -HUP inetd`. Die verfügbaren RPC-Dienste werden mit `rpcinfo -p` ausgegeben, und für alle anderen

Zusammenfassung

Die Abhärtung von Linux-/BSD-Systemen (und ähnlichen unixartigen Systemen) kann teilweise recht einfach realisiert werden, kostet allerdings verhältnismäßig viel Zeit und ist sehr aufwendig, wenn man dies für eine ganze Reihe von Systemen erledigen möchte. Beispielsweise muss bei jedem Kernel-Update ein neuer Kernel gepatcht und kompiliert werden, weil Standarddistributionen keinen gehärteten GRSecurity-Kernel beinhalten. Abhilfe schaffen an dieser Stelle einfache Updates von Kernel-Packages in den vorgestellten Security-Distributionen (bzw. alternativ die Verwendung von OpenBSD). Diese verfügen meist auch ohne weiteres Zutun des Anwenders über die in diesem Artikel besprochenen Sicherheitsvorkehrungen. ●

Über den Autor

Steffen Wendzel beschäftigt sich seit vielen Jahren mit der Sicherheit von Unix-Systemen und TCP/IP-Netzwerken. Er entwickelte diverse OpenSource Software, ist Maintainer der Hardened Linux Distribution, Autor mehrerer Bücher zu den Themen Linux und Netzwerksicherheit, Security Consultant und Seminarleiter bei Plötner-IT sowie Student der Informatik an der FH-Kempten (Deutschland). Seine Webseite: <http://cdp.doomed-reality.org>.

Links und Literatur

- S. Wendzel & J. Plötner: Praxisbuch Netzwerksicherheit, Galileo-Press, 2. Auflage, Februar 2007;
- <http://www.ploetner-it.de/downloads/slac-Hardening.pdf>, S. Wendzel: Hostbasierte Sicherheit und Serverhärtung;
- M. Piotrowski: Festung Linux – eine Übersicht von Projekten, Hakin9 2/06;
- http://www.trapkit.de/papers/ssp_v1_20031004.pdf, T. Klein: Stack Smashing Protector;
- Jon Erickson: Forbidden Code, mitp;
- <http://pax.grsecurity.net>, PaX;
- <http://grsecurity.net>, GRSecurity;
- <http://grsecurity.net/quickstart.pdf>, Introduction to GRSecurity;
- <http://www.trl.ibm.com/projects/security/ssp/>, GCC ProPolice Patch;
- <http://hardenedlinux.sf.net>, Hardened Linux;
- <http://www.openwall.com/Owl/>, OpenWall;
- <http://www.adamantix.org/>, Adamantix;
- P. H. Gregory: Solaris Security, Prentice Hall PTR, August 1999;
- M. Herrb: Enchancing Xfree86 Security, July 2003.