Mit DVD

iX KOMPAKT ADMINISTRATION

Ein Sonderheft des Magazins für professionelle Informationstechnik, www.ix.de 3/2014

Auf der Heft-DVD

Fertige VMs zum Ausprobieren: Debian mit Bacula 7, CentOS mit Icinga 2 und Icinga Web 2, UCS 3.2-2 (25 User)

Software, Tools, Doku: FreeNAS 9.2.1.5, nützliche Administrations-Helferlein für Linux, Mac OS X und Windows, Kerberos-Key-Management, Netz-Tools, RFCs u. v. a. m.

Open Source im Einsatz

Systemkonfiguration:

Backup mit Bacula 7

Do-it-yourself-NAS

Sicherheit:

Compliance-Management mit OpenSCAP

SELinux und Grsecurity effektiv nutzen

Netzwerk:

Traffic-Analyse mit Wireshark

WLAN mit RADIUS

Monitoring:

Nagios und Icinga 2

Logdatenanalyse

Linux-AD-Integration:

FreeIPA-Tutorial

Kerberos – Tools und Tipps

Cloud-Tutorial:

Mit OpenStack zur eigenen Private Cloud

Managed Hosting

zertifiziert nach ISO 27001 und ISO 9001

15 Jahre Erfahrung in Managed Hosting und Open Source

Individuelles Hosting am Standort Frankfurt/Main mit persönlichem und kompetentem Support.

Professionelle Hostinglösungen vom Server bis zum Cluster-Cloudsystem mit Beratung, Planung und Service 24/7.

Wir bieten über 15 Jahre Erfahrung in Open Source, Systemadministration und Managed Hosting.
Für mehr Performance, Sicherheit und Verfügbarkeit.

hostserver.de/hosting

HOSTSERVER
Berlin ▪ Marburg ▪ Frankfurt am Main

Beratung unter:
0 30 / 47 37 55 50

Hosting made in Germany

Cat-Content satt

Nach landläufiger Meinung soll beim sogenannten Cat-Content angeblich kein Sysadmin widerstehen können – egal ob Cat-5e oder lebende Miezekätzchen. Während sich Letztere in diesem *iX*-Kompakt zu Administration mit Open-Source-Tools als Aufmacherbilder tummeln, spielen Cat-*x*-Kabel als Transportmedium im Tagesgeschäft der IT eine tragende Rolle. Sie sind quasi die Nervenenden des globalen Rückgrats Internet, über das Anwender und Anwendungen Daten austauschen.

Im Internet funktionierten von Anfang an große Teile der Infrastruktur mit Open-Source-Betriebssystemen und -Programmen – etwa Linux oder der Webserver Apache. In anderen Bereichen war freie Software zunächst überwiegend technikaffinen und experimentierfreudigen Naturen vorbehalten.

Doch inzwischen hat das dahinter stehende Entwicklungsmodell große Verbreitung und Akzeptanz gefunden. Selbst klassische IT-Unternehmen wie HP, IBM und Co. setzen für ihre Angebote teilweise – Tendenz steigend – auf freie Komponenten. Die wiederum müssen in vielen Bereichen den Leistungsvergleich mit ihren kommerziellen Gegenstücken nicht scheuen, laufen ihnen zum Teil sogar den Rang ab.

Ein aktuelles Beispiel hierfür ist OpenStack. Das ursprünglich von der NASA und der amerikanischen Rackspace Inc. initiierte Projekt hat sich mit mittlerweile rund 200 unterstützenden Firmen zum Quasi-Standard für offene Cloud-Angebote gemausert. Allein die Liste der Gold- und Platin-Mitglieder liest sich wie das Who-is-Who der IT-Branche. Dabei setzen produktseitig nicht nur die „üblichen Verdächtigen“ wie Red Hat, SUSE oder Canonical auf OpenStack, auch HP und IBM haben darauf basierende Cloud-Angebote im Portfolio. Apropos: Ein dreiteiliges Tutorial in diesem Heft zeigt Schritt für Schritt, wie sich das aktuelle OpenStack „Icehouse“ im eigenen Umfeld einsetzen lässt.

Aber nicht nur bei den Infrastrukturkomponenten kann Open Source punkten, auch beim Administrieren und Steuern der zugehörigen Systeme existieren neben dem breiten Strauß kommerzieller Pakete leistungsfähige freie Alternativen. Letztlich kommen vor allem in komplexen Umgebungen die IT-Verantwortlichen nicht ohne effiziente Werkzeuge für die Systemverwaltung aus. Hier kann die Community ein vielfältiges Angebot vorweisen. Gerade im Bereich Monitoring setzt man derzeit quasi die Standards dessen, was machbar ist.

Jedoch fordert die Komplexität der Anwendungen ihren Tribut: Die gängige Vorstellung, Open-Source-Software koste ja nichts, stimmt nur bedingt. Zwar fallen keine Lizenzkosten an, aber für einen wirkungsvollen Einsatz muss man entweder in das Know-how seiner Mitarbeiter investieren oder externes Wissen von Dienstleistern einkaufen. Der Gewinn, der in jedem Fall bleibt, ist die Offenheit der eingesetzten Tools.

ANDRÉ VON RAISON

Eine Frage der richtigen Überwachung

Wem seine IT am Herzen liegt, der kümmert sich nicht erst bei Fehlern um das Wohlbefinden der Systeme. Die Open-Source-Pakete Nagios und das daraus entstandene Icinga sind heute die verbreitetsten Anwendungen für das Monitoring. Wie man mit diesen und weiteren Tools seine Umgebung automatisch erfasst sowie sie sicher im Auge behält

ab Seite 81

Sicherheit

Identitätsmanagement

Systemkonfiguration

Monitoring

Netzwerk

Cloud und mehr

Im Open-Source-Cloud-Umfeld hat sich das Projekt OpenStack mittlerweile zum De-facto-Standard entwickelt – die Zahl von über 200 unterstützenden Unternehmen spricht für sich. Ein Tutorial beschreibt Schritt für Schritt den Weg zur eigenen „Private Cloud“.

ab Seite 135

Wer ist wer

Früher oder später stellt sich in gemischten Umgebungen die Frage, wie man die Benutzerverwaltung von unixoiden und Windows-Systemen unter einen Hut bekommt. Für das Themenfeld Identity-Management bietet die Open-Source-Community leistungsfähige Alternativen.

ab Seite 33

Virtualisierung und Cloud

Sonstiges

Auf der Heft-DVD

Software zum Heft: Bacula 7.0.3, FreeNAS 9.2.1.5, Kerberos-Tools (kstart 4.1, msktutil 0.5.1, realmd 0.14.5, SSSD 1.11.5.1), Logstash (Elasticsearch 1.1.1, Kibana 3.0.1, Redis 2.8.9), Octopussy 1.0.14, OpenSSL 1.0.1g (Windows), WinSCP 5.3.3, Wireshark 1.10.7, Zenmap 6.46

Weitere Tools: 7-Zip, dd_rescue 1.40, dd_rhelp 0.1.2, EtherAPE 0.9.13, Iperf 2.0.5/3.0, liboping 1.6.2, Ncat 6.46, Ndisc6 0.7.1, Netperf 2.6.0, Nmap 6.46, Nping 0.6.46, OpenVPN 2.3.3, OpenVPN GUI 1.0.3, p7zip 9.20.1, Putty, tinc 1.0.24, Tunnelblick 3.1.7

Dokumentation: M23-Handbuch, OpenLDAP Admin Guide, Wireshark User Guide, über 350 RFCs

Fertige VMs zum Ausprobieren:
- Debian-Wheezy mit Bacula 7
- CentOS 6.5 mit Icinga 2 und Icinga Web 2
- Univention Corporate Server 3.2-2 (25 User)

Hinweis für Käufer der digitalen Ausgaben
- PDF- und iPad-Version: In der *iX*-App finden Sie einen Button zum Download des DVD-Images.
- PDF-E-Book: Folgen Sie im Browser der unter „Alle Links“ angegebenen URL zum DVD-Image.

Alle Links: www.ix.de/ix1416004

Alle Links: www.ix.de/ix1416SSS Artikel mit Verweisen ins Web enthalten am Ende einen Hinweis darauf, dass diese Webadressen auf dem Server der *iX* abrufbar sind. Dazu gibt man den *iX*-Link in der URL-Zeile des Browsers ein. Dann kann man auch die längsten Links bequem mit einem Klick ansteuern. Alternativ steht oben rechts auf der *iX*-Homepage ein Eingabefeld zur Verfügung.

MAGAZIN FÜR PROFESSIONELLE INFORMATIONSTECHNIK

***iX* Kompakt 3/2014 – Administration**

Postfach 61 04 07, 30604 Hannover; Karl-Wiechert-Allee 10, 30625 Hannover

Redaktion: Telefon: 05 11/ 53 52-387, Fax: 05 11/ 53 52-361, E-Mail: post@ix.de

Chefredakteur: Jürgen Seeger (js) -386

Konzeption und redaktionelle Leitung: André von Raison (avr) -377, E-Mail: avr@ix.de

Ständige Mitarbeiterin: Barbara Lange

Autoren dieser Ausgabe: Christoph Arnold, Christian Berendt, Peter Fetzer, Oliver B. Fischer, Alexandros Gougousoudis, Safuat Hamdy, Peer Heinlein, Patrick Preuß, Mark Pröhl, André von Raison, Michael Riepe, Thorsten Scherf, Andreas Schmidt, Norbert Tretkowski, Tilman Wittenhorst, Erkan Yanar

Abbildungen © Can Stock Photo Inc.: Siberia (Titel, DVD, DVD-Hülle), fikmik (S. 4, 33), Kirill (S. 4, 81), myrainjom01 (S. 4), taden (S. 4, 135), vladstar (S. 7), cfoto (S. 8), cynoclub (S. 16, 72), ESIGHT (S. 24), jgroup (S. 28), andreykuzmin (S. 34, 40, 46) 2002Lubava1981 (S. 52), vitalytitov (S. 58), Pakhnyushchyy S. 61), bloodua (S. 62), McIninch (S. 67), ifong (S. 72), oxilixo (S. 76), pterwort (S. 90), tobkatrina (S. 96), w20er (S. 105), maigi (S. 106), McIninch (S. 112), pontuse (S. 128), flibustier (S. 136, 159), Meldes (S. 142), Colecanstock (S. 148), Svet_lana (S. 155),
Abbildungen © Fotolia.com: Eric Isselée (S. 82, 117, 120), Ana Gram (S. 118), Patryk Kosmider (S. 124)

Redaktionsassistenz: Carmen Lehmann (cle) -387, Michael Mentzel (mm) -153

Korrektorat: Silke Peters, Hinstorff Verlag, Rostock

Layout und Satz: Enrico Eisert, Kathleen Tiede, Matthias Timm, Hinstorff Verlag, Rostock; Jürgen Gonnermann, Heise Zeitschriften Verlag

Titelidee: *iX*, André von Raison

Verlag: Heise Zeitschriften Verlag GmbH & Co. KG, Postfach 61 04 07, 30604 Hannover; Karl-Wiechert-Allee 10, 30625 Hannover; Telefon: 05 11/53 52-0, Telefax: 05 11/53 52-129

Geschäftsführer: Ansgar Heise, Dr. Alfons Schräder

Mitglied der Geschäftsleitung: Beate Gerold

Verlagsleiter: Dr. Alfons Schräder

Anzeigenleitung: Michael Hanke (-167), E-Mail: michael.hanke@heise.de

Druck: Dierichs Druck + Media GmbH, Kassel

Verantwortlich: Textteil: Jürgen Seeger; Anzeigenteil: Michael Hanke

***iX* Kompakt 3/2014 – Administration:** Einzelpreis € 12,90, Österreich € 14,20, Schweiz sfr 25,80, BeNeLux: € 14,80, Italien: € 16,80

Die Inserenten

Die hier abgedruckten Seitenzahlen sind nicht verbindlich. Redaktionelle Gründe können Änderungen erforderlich machen.

Demo-Appliance mit Univention Corporate Server

Die Heft-DVD enthält eine vorinstallierte und -konfigurierte virtuelle Appliance mit dem Univention Corporate Server (UCS) in der Version 3.2-2. Sie umfasst eine bis 30.6.2015 gültige Lizenz für bis zu 25 Benutzer für Testzwecke und den Einsatz im privaten, nicht kommerziellen Bereich. Mit der Appliance lassen sich komplexe Serverstrukturen mit Master-, Slave- und Backup-Servern aufsetzen, Nutzer und Gruppenberechtigungen anlegen sowie zusätzliche Enterprise-Apps installieren und verwalten.

Die virtuelle Appliance ist auf „Bridge Mode" eingestellt, sodass das UCS-System über das lokale Netzwerk erreichbar ist. Nach dem Entpacken des VMware-Images, Laden der *vmx*-Datei in den VMware Player und Starten der virtuellen Maschine erscheint das Boot-Menü. Nach einigen Sekunden fährt dessen vorausgewählter Eintrag „Univention Corporate Server" (UCS) hoch, und ein Einrichtungsassistent führt Schritt für Schritt durch die Inbetriebnahme.

Als erster Schritt folgt die Auswahl der Systemrolle. Bei der Erstinstallation von UCS ist der vorausgewählte „Domaincontroller Master" zu verwenden, Sprach-, Tastatur- und Zeitzoneneinstellungen schließen sich an. Danach fragt der Assistent den Rechnernamen inklusive Domäne ab und generiert daraus automatisch einen Windows-Domänennamen.

Diesen Namen kann man später nicht mehr ändern!

Zum Test einer Active-Directory-(AD)-Migration muss der Windows-Domänenname der zu migrierenden AD-Domäne entsprechen (siehe „Alle Links"). Rechnername und IP-Adresse des UCS-Systems müssen sich hingegen vom AD-Server unterscheiden. Das festzulegende Root-Passwort gilt automatisch auch für das Administrator-Konto.

Zwar ist die Netzkonfiguration auf DHCP voreingestellt, man kann aber auch eine statische Adresse vergeben. In den restlichen Abschnitten kann man die vorgeschlagenen Einstellungen übernehmen. Je nach Testzweck lassen sich nun weitere Software-Komponenten wie Samba 4 als AD-kompatibler Domaincontroller auswählen oder später über das App Center nachinstallieren.

Remote-Zugriff per SSH

Im Anschluss daran nimmt das Einrichten der Domäne und die Installation der zusätzlichen Softwarekomponenten einige Minuten in Anspruch. Nach einem Neustart ist das UCS-System einsatzbereit. Der Remote-Zugriff kann entweder per Secure Shell (SSH) als Benutzer „root" mit dem vergebenen Passwort oder im Web-Managementsystem als „Administrator" erfolgen. Hat man die KDE-Desktop-Umgebung installiert, kann man sich dort ebenfalls mit dem Administrator-Konto anmelden.

UCS kann durch die Integration von Samba 4 als AD-kompatibler Domänen-Controller fungieren und Benutzer, Drucker, Rechner sowie Richtlinien verwalten. Mit dem Beitritt eines Windows Client zur UCS-Domäne lässt sich der Funktionsumfang von UCS am einfachsten testen. Weitere Testszenarien sind mit dem App Center schnell aufgebaut, beispielsweise zum Evaluieren von Migrationsszenarien oder einzelner Business-Applikationen wie Groupware oder File-Sharing-Ansätze.

Sicherheit: Flatterhafte Werte

Sensible Informationen verschwinden oft schneller aus IT-Systemen als manchem lieb ist. Die für die Zugangsberechtigungen verantwortlichen Administratoren fühlen sich hier schnell ans sprichwörtliche Flöhe hüten erinnert. Wer aber ein bisschen Aufwand nicht scheut, kann sich mit weniger flüchtigen Datenschätzchen belohnen.

Linux-Systeme härten mit Grsecurity

Abgrenzungshilfe

Safuat Hamdy

Für die Härtung von Linux-Systemen steht Grsecurity in einer Reihe mit anderen Frameworks wie AppArmor, SELinux, SMACK oder TOMOYO. Es unterscheidet sich von diesen jedoch nicht nur in Details, sondern hebt sich grundsätzlich durch einige spannende Features ab.

Obwohl der konzeptionelle Mehrwert von Security-Frameworks wie AppArmor, SELinux, SMACK oder TOMOYO und anderen unbestritten ist, finden diese in der Praxis jedoch kaum Anwendung. Das Haupthindernis ist oft der mit dem Einstieg verbundene Lernaufwand, vor allem beim Erstellen einer brauchbaren Policy. Dieser Artikel zeigt, wie der Einstieg in Grsecurity mit relativ moderatem Aufwand gelingt.

Der Betrieb exponierter Systeme wie Webserver oder Mail-Relays stellt aus Sicherheitssicht nach wie vor eine besondere Herausforderung dar. Solche Systeme sind auf verschiedenen Ebenen angreifbar. Dies gilt besonders für Webserver mit komplexen Webanwendungen. Kritische Schwachstellen könnten dazu führen, dass Angreifer Dienste oder gleich das gesamte System beeinträchtigen und schlimmstenfalls übernehmen können. Infiziert ein Hacker im Zuge eines solchen Angriffs ein System mit einem Rootkit, könnte es Teil eines Botnetzes und anschließend für weitere illegale Aktivitäten missbraucht werden.

Ein prominentes Beispiel hierfür ist das Ende 2011 bekannt gewordene Rootkit Ebury. Hierbei ersetzten die Angreifer im Zuge der Kompromittierung das *sshd*-Binary und weitere Programme. In der Folge dienten die befallenen Systeme dazu, Zugangsdaten zu erbeuten, Spam zu versenden und Websurfer auf Seiten umzuleiten, die Malware über Drive-by-Downloads verteilen (siehe „Onlinequellen“, [a]).

Systemhärtung – eine Begriffsabgrenzung

Allgemein sind Betreiber dazu angehalten, ihre Systeme zu härten, doch ist dieser Begriff etwas schwammig. Als erstes fällt da das Abschalten unnötiger Dienste und das Deinstallieren unnötiger Software ein, um die Angriffsoberfläche zu verkleinern. Diese an sich gute Idee kann sich aber als trickreich erweisen, da viele Pakete in einem komplexen Geflecht aus Abhängigkeiten verstrickt sind. Daneben stehen Standardmaßnahmen wie der Einsatz von Paketfiltern und Anti-Viren-Software, das Einschränken von Zugriffsrechten, ein restriktives Handhaben von *su* und *sudo*, Passwörter für den Bootloader, ein Reduzieren der Programme mit SUID-Bit auf das absolut Notwendige et cetera. Im Netz existieren verschiedene ausführliche Härtungs-Guides wie die Benchmarks des Center for Internet Security [b], die ein gewissenhafter Administrator Punkt für Punkt abarbeiten kann.

Doch diese Maßnahmen kratzen nur an der Oberfläche und bleiben auf bestenfalls halbem Weg zu einem robusten System stehen. Viele Einzelmaßnahmen gehen am Punkt vorbei. Nimmt man den Fall Ebury, dann erfolgte die Kompromittierung offenbar über das Abfangen von Root-Passwörtern an anderer Stelle – da hilft es nur wenig, seine (auf Servern nicht so zahlreich vorhandenen) Benutzer einzuschränken, rigide Passwort-Richtlinien durchzusetzen und die Anzahl der SUID-root-Programme auf ein Minimum zu reduzieren.

Andere Malware setzt oft auf Zero-Day-Exploits, um sich Zugang zu Systemen zu verschaffen. Hierbei nutzt sie in der Öffentlichkeit noch unbekannte Schwachstellen aus. Auch da haben die oben genannten Maßnahmen oft nur die Wirkung, die Angriffsoberfläche zu verkleinern. Das ist nicht verkehrt, aber Härtung ist mehr, als einem Angreifer nur auszuweichen – letztlich kann man die Angriffsoberfläche nicht beliebig verkleinern. Für einen bestimmten Zweck im Internet betriebene Systeme müssen einem Angriff auch widerstehen können.

Kaum mehr als ein Feigenblatt: *chroot*

Administratoren von Unix-Systemen preisen *chroot*-Umgebungen immer wieder gern als Maßnahme zur Erhaltung der Systemintegrität. Diese Umgebungen sind aber nicht wasserdicht, es sind im Lauf der Zeit verschiedene Ausbruchsmethoden bekannt geworden. Tatsächlich ist *chroot* nicht als Sicherheitsmechanismus entstanden, sondern nur als Mittel, die Sicht auf das Dateisystem einzuschränken. Zudem ist das korrekte Aufsetzen einer *chroot*-Umgebung nicht trivial.

Eine Weiterentwicklung von *chroot*-Umgebungen besteht in Containern wie OpenVZ oder Linux Containern (LXC) siehe Seite 155. In einem solchen Container bekommen die darin enthaltenen Prozesse nicht nur eine eingeschränkte Sicht auf das Dateisystem, sondern auf fast alle Aspekte des Betriebssystems, etwa eigene Prozesstabellen, Netz-Interfaces mit eigenen IP-Adressen und Firewall-Regeln et cetera. Aber auch hier gibt es Fallstricke – LXC beispielsweise war bis vor kurzem nicht in der Lage, einen Superuser innerhalb eines Containers von dem Superuser des Wirts abzugrenzen.

RBAC und MAC – erweiterte Zugriffskontrolle

Eine konzeptionell ausgezeichnete Maßnahme gegen eine große Klasse noch unbekannter Schwachstellen besteht in rollenbasierter Zugriffskontrolle (Role Based Access Control, RBAC) im Zusammenspiel mit richtliniengesteuerter Zugriffskontrolle (Mandatory Access Control, MAC). RBAC beruht im Wesentlichen darauf, verschiedene Berechtigungen in Rollen zu bündeln und diese dann kollektiv Benutzern beziehungsweise Prozessen zu gewähren. Oberflächlich betrachtet ist dies dem Gruppenkonzept recht ähnlich, tatsächlich besteht aber ein konzeptioneller Unterschied, da Gruppen Benutzer bündeln, denen man kollektiv einzelne Rechte gewähren kann.

MAC beruht darauf, Zugriffe über eine zentrale Richtlinie zu steuern, auf die die einzelnen Benutzer oder Prozesse keinen Einfluss haben. Das unter Unix und Linux hierfür übliche Modell ist identitätsbasierte Zugriffskontrolle (Identity Based Access Control, IBAC) im Zusammenspiel mit benutzergesteuerter Zugriffskontrolle (Discretionary Access Control, DAC), das heißt, man muss den Linux-Kernel für RBAC und MAC erst um die entsprechenden Funktionen erweitern.

Im Lauf der Zeit integrierten die Entwickler verschiedene Frameworks für RBAC und MAC in den Linux-Kernel, allen voran SELinux, von dem es inzwischen sogar Portierungen zu BSD (SEBSD) und Mac OS (SEDarwin) gibt. Das Erstellen von

SELinux-Richtlinien erfordert jedoch eine hohe Kompetenz, was de facto eine breitere Nutzung von SELinux verhindert. Die beiden Artikel ab Seite 24 und 28 wollen dazu beitragen, den Kreis der SELinux-User zu erweitern. Andere Frameworks wie AppArmor, SMACK oder TOMOYO sind mit dem Anspruch angetreten, relativ leicht nutzbar zu sein, jedoch zeigt die Praxis, dass auch hier der Teufel im Detail steckt.

Ein Einsatz erfordert meist Handarbeit

Ein weiteres Framework für RBAC und MAC wurde in Grsecurity implementiert. Grsecurity bietet allerdings weit mehr als RBAC und MAC, aber es ist kein Bestandteil des offiziellen Kernels, was die Nutzung leider zusätzlich erschwert. Wie es aussieht, wird sich daran auch nichts ändern [c]. Für einen Betreiber liegt die Hürde zur Nutzung von Grsecurity damit recht hoch. Dies ist recht bedauerlich, denn Grsecurity kann mit Fug und Recht als ein gelungener Beitrag zur Linux-Sicherheit bezeichnet werden. Zudem ist Grsecurity einfacher zu handhaben als SELinux [d, e].

Unter den genannten Frameworks zur Linux-Härtung steht Grsecurity etwas abseits, da es nicht die LSM-Schnittstelle (Linux Security Modules) nutzt, sondern aus einem Satz von Patches besteht, der viele Bereiche des Linux-Kernels berührt. Diese muss der Nutzer eigenhändig installieren und den Kernel kompilieren. Das ist nicht jedermanns Sache. Es gibt leider nur wenige Distributionen wie Hardened Gentoo, bei denen die Maintainer die Integration übernehmen. Für Projekte wie Debian oder Ubuntu stehen zumindest inoffizielle Kernel-Pakete mit Grsecurity zur Verfügung (siehe „Alle Links“). Auch Arch Linux und Linux from Scratch bieten einen Grsecurity-gepatchten Kernel an, ebenso wie die Slackware-Variante SlackPax, die aber anscheinend seit 2013 nicht so recht gepflegt wird. Die „großen“ kommerziellen Distributionen wie RHEL, SLES oder Mandriva kommen dagegen höchstens mit integriertem SELinux daher.

Es gibt viele Gründe dafür, dass Grsecurity LSM nicht nutzt. Abgesehen davon, dass der Hauptentwickler von Grsecurity, Brad Spengler, eine tiefe Abneigung gegen LSM pflegt, gibt es den ganz pragmatischen Grund: Der Funktionsumfang von Grsecurity geht weit über die Möglichkeiten von LSM hinaus (siehe „Alle Links“). So enthält Grsecurity Maßnahmen zum Abdichten von *chroot*-Umgebungen und schränkt den Zugang zum Kernel über */dev/mem*, */dev/kmem* et cetera erheblich ein. Darüber hinaus ist PaX ein Bestandteil von Grsecurity. PaX bringt Härtung gegen diverse Speicherprobleme mit sich, unter anderem Address Space Layout Randomization (ASLR, ein Begriff, den das PaX-Projekt geprägt hat), und verhindert die Ausführung von schreibbaren Speicherbereichen (auch bekannt als Data Execution Prevention, DEP). Weitere Grsecurity-Features bestehen in Trusted Path Execution (TPE) zum Verhindern der Ausführung von Trojanern, Einschränkungen bei der Sicht auf die Prozesstabelle sowie beim *ptrace*-Systemaufruf und viele andere. Eine umfassende und übersichtliche Beschreibung aller Features findet sich auf der Website des Projekts [f]. Hauptentwickler Spengler hat dort auch ein gutes Tutorial zur Einführung in RBAC hinterlegt [g].

Listing 1: User-bezogener Abschnitt der Policy

```
role root uG
role_transitions admin shutdown
role_allow_ip   127.0.0.6/32
role_allow_ip   0.0.0.0/32

...

# Role: root
subject /bin/ping6 o {
   /                            h
   /bin                         h
   /bin/ping6                   x
   /etc                         h
   /etc/host.conf               r
   /etc/hosts                   r
   /etc/ld.so.cache             r
   /etc/nsswitch.conf           r
   /etc/resolv.conf             r
   /lib                         rx
   /lib/modules                 h
   /usr                         h
   /usr/lib/libcrypto.so.1.0.0  rx
   -CAP_ALL
   +CAP_NET_RAW
   bind                         disabled
   connect                      disabled
   sock_allow_family            ipv6
}
```

Grsecurity – ein praktischer Einstieg

Da Grsecurity wie schon erwähnt nicht Bestandteil des offiziellen Kernels ist, muss man die Patches von Hand installieren, den Kernel konfigurieren und übersetzen. Eine Schwierigkeit dabei kann darin liegen, dass die Patches für den sogenannten „Vanilla“-Kernel-Quellcode gelten; ein Konflikt mit den modifizierten Quellen einzelner Distributionen ist denkbar. Als Betreiber riskiert man beim Einsatz eines inoffiziellen Kernels zudem den Verlust des Supports von Distributionen wie RHEL oder SLES. Wer sich auf das (lohnende) Unternehmen Grsecurity einlässt, sollte daher besser professionellen Support hinzuziehen.

Nun kommt der eigentlich spannende Teil, wie man eine RBAC-Richtlinie erstellt. Eine Beschreibung findet man im Wiki zu Grsecurity [h]. Dessen Pflege ist allerdings nicht optimal, weshalb man im Zweifelsfall die ausführliche Beschreibung in der mitgelieferten Policy heranziehen muss. Ist alles installiert und einsatzbereit, gibt es zwei Möglichkeiten: Entweder startet man ausgehend von der zum Softwareumfang gehörenden Policy oder betreibt Grsecurity im vollen Lernmodus.

Über Letzteren kann sich ein Administrator eine Policy erstellen. Man unterscheidet zwischen Full System Learning sowie rollen- und subjektspezifischem Learning (siehe unten). Ersteres berücksichtigt alle Systemaktivitäten für den Lernprozess, das spezifische Learning zieht nur die Aktivitäten bestimmter Rollen oder Subjekte heran. Der Lernprozess erfolgt so, dass das System alle Zugriffe und die dazugehörigen Parameter protokolliert. Nach dem Beenden des Lernmodus lässt sich aus dem Protokoll eine Policy erzeugen. Die befindet sich in */etc/grsec/policy*, dieser Pfad ist fest vorgegeben. Full System Learning startet der Superuser mit dem Kommando

```
gradm -F -L /etc/grsec/learning.log
```

Zuvor muss er aber einige Passwörter setzen, nämlich für Grsecurity an sich, für die Rolle „admin“ und für die Rolle „shutdown“. Dies bewerkstelligen die Kommandos

```
gradm -P
gradm -P admin
gradm -P shutdown
```

Die Passwörter (das heißt deren Hash-Werte) speichert *gradm* in der Datei */etc/grsec/pw*. Ohne diese weigert sich Grsecurity (genauer die RBAC-Komponente), in Aktion zu treten. Hintergrund: Grsecurity wird (wenn die Policy vorliegt) mit dem Kommando *gradm -E* scharf geschaltet und per *gradm -D* deaktiviert. Letzteres – auch aus dem Lernmodus – erfordert das Grsecurity-Passwort. Für administrative Tätigkeiten muss man in die Rolle *admin* wechseln.

Was wir herstellen, das testen wir. Was wir testen, das läuft.

Um zu gewährleisten, dass unsere Systeme sich völlig problemlos in Ihre Serverinfrastruktur implementieren lassen und dort reibungsfrei arbeiten, haben wir neben diversen technischen Features ein spezielles Hardware-Testgerät entwickelt: Wir nennen es „Thomas-Krenn-Techniker“. Wir bieten nichts an, was wir nicht getestet haben. Das nennen wir Service made in Germany. thomas-krenn.com/thomas-krenn

Open Source Förderung 2014

Auch dieses Jahr prämierten wir herausragende Projekte im Rahmen der Open Source Förderung mit Hardware-Gewinnen im Gesamtwert von 6.500 €!

Lesen Sie mehr zum Thema Open Source unter thomas-krenn.com/projekte_2014

THOMAS KRENN®

server.hosting.customized.

Für den Lernmodus sind zwei Dinge sehr wichtig: Zum Einen darf man darin keine administrativen Tätigkeiten ausführen. Sonst lernt das System, dass privilegierte Operationen auch in den gewöhnlichen Nutzungsszenarien vorkommen dürfen und die daraus abgeleitete Policy fällt viel zu großzügig aus. Zum anderen sollte man das System für eine gewisse Dauer so benutzen, dass alle legitimen Nutzungsszenarien mehrfach durchgespielt werden. Hierzu gehören alle Formen der Anmeldung am System, sei es über SSH oder auch über eine virtuelle Konsole.

Ist man zu der Ansicht gelangt, dass die Trainingsphase alle Nutzungsszenarien abgedeckt hat, kann man Grsecurity mit *gradm -D* deaktivieren und den Lernmodus beenden. Ruft man *gradm* erneut im Lernmodus auf, hängt Grsecurity die Protokolleinträge an die mit dem Parameter *-L* spezifizierte Datei an.

Die ersten Schritte zur Policy

Für das Bilden einer Policy nutzt man nun *gradm -F -L /etc/grsec/learning.log -O /etc/grsec/newpolicy*. Diese Policy ist in Bezug auf die im Lernmodus durchgeführten Aktionen äußerst restriktiv. Für jeden im Lernmodus aktiven Benutzer legt Grsecurity eine eigene Benutzerrolle an. Hat beispielsweise der User Root */bin/ping6* aufgerufen, dann erscheint in der Policy ein Ausschnitt wie in Listing 1 gezeigt.

Der Ausschnitt besagt: Es gibt eine Rolle *root*; diese ist eine Benutzerrolle – zu erkennen an dem Rollenmodus-Flag *u* –, das heißt, jeder Prozess mit der tatsächlichen UID von Root erhält sie automatisch. Die effektive UID eines Prozesses ist hierfür irrelevant, der Wechsel in eine andere Benutzerrolle erfolgt nur dann, wenn der entsprechende Prozess den Systemaufruf *setuid()* verwendet. Neben Benutzerrollen gibt es noch Gruppenrollen sowie spezielle Rollen. Gruppenrollen sind durch den Modus *g* gekennzeichnet und funktionieren analog zu User-Rollen. Ist keine Benutzer- oder Gruppenrolle für einen Benutzer anwendbar, kommt die Default-Rolle zum Einsatz. Gibt es aufgrund der UID und der GID mehrere Möglichkeiten für eine Rollenzuweisung, hat die Benutzerrolle Vorrang vor der Gruppenrolle.

Spezielle Rollen sind durch *s* gekennzeichnet und keiner UID oder GID zugewiesen. Zum Wechseln in diese Rollen muss man per *gradm* eine Rollen-Transition veranlassen. Dies erfordert in der Regel eine Authentisierung, man muss für die Rolle also ein Passwort spezifizieren. Diese Transitionen können aber auch mit Authentisierung nicht willkürlich erfolgen. Zunächst muss für eine Rolle das Nutzen von *gradm* und des dahinterstehenden Mechanismus zulässig sein. Dies kennzeichnet der Rollenmodus *G*. Außerdem sind alle zulässigen Transitionen in der Richtlinie aufzuführen – per Rollenattribut *role_transition*. Im Beispiel von Listing 1 darf ein Nutzer aus der Rolle *root* in die (nicht weiter aufgeführten) speziellen Rollen *admin* und *shutdown* wechseln.

Ein weiteres Rollenattribut ist *role_allow_ip*. Es schreibt vor, dass man eine Benutzerrolle (hier *root*) nur dann einnehmen kann, wenn man sich von einer der unter diesem Attribut vermerkten IP-Adressen aus angemeldet hat. Dies dürfte man in der Regel anpassen wollen, da der Lernmodus in vielen Fällen hierfür kein sinnvolles Ergebnis erzielen kann.

Zu jeder Rolle gehört die Spezifikation einer Reihe von Subjekten – entweder Programme oder Verzeichnisse. Letzteres fasst alle darunter im Dateisystembaum stehenden Programme zusammen. Das Beispiel zeigt die Subjekt-Spezifikation für */bin/ping6*. Dabei folgen dem Programmnamen die Subjektmodus-Flags, hier der Modus *o* (override). Zu den Subjektmodi später mehr. Der interessanteste Teil ist die Objektspezifikation. Sie legt fest, auf welche Dateien das Subjekt wie zugreifen darf, markiert durch den Objektmodus. Neben den „traditionellen" Zugriffsarten *r* (read), *w* (write) und *x* (execute) nimmt Grsecurity eine weitere Differenzierung vor. So gibt es noch *a* (append), *c* (create), *d* (delete), *l* (link) und *h* (hide), wobei Letzteres eine Zugriffseinschränkung bedeutet – die so markierten Objekte sind für das Subjekt schlicht unsichtbar.

Von Subjekten, Objekten und ihren Flags

Ein Objekt kann eine einzelne Datei oder ein Verzeichnis sein. Die Angabe eines Verzeichnisses bewirkt das rekursive Anwenden des Objektmodus auf alle inner- und unterhalb liegenden Dateien und Verzeichnisse. Für das Spezifizieren mehrerer Objekte sind „*" und „?" als Wildcards zulässig. Für die Zugriffsregeln auf eine Datei gilt, dass ein spezifischerer Pfad Vorrang vor einem unspezifischeren hat. So ist eine Regel für */bin/ping6* spezifischer als eine für */bin* oder /, aber auch als eine für */bin/**.

Listing 2: Richtlinienausschnitt zur Vererbung

```
subject / {
    /                   h
    /bin                rx
    /usr/bin            rx
    /lib                rx
    /lib/modules        h
    /usr/lib            rx
    ...
    -CAP_ALL
    +CAP_DAC_READ_SEARCH
}

subject /bin/foo {
    /usr/libexec        rx
    -CAP_DAC_READ_SEARCH
    +CAP_NET_ADMIN
}
```

Listing 3: Zu stark eingeschränkte Rechte

```
subject /bin/rm o {
    /                           h
    /bin                        h
    /bin/rm                     x
    /etc                        h
    /etc/ld.so.cache            r
    /lib                        h
    /lib/ld-2.15.so             x
    /lib/libc-2.15.so           rx
    /var                        h
    /var/spool/cron/lastrun/lock   wd
    -CAP_ALL
    bind    disabled
    connect disabled
}
```

Listing 4: Beispiel Rechte übertragen

```
subject /sbin/mydaemon {
    /                   h
    /bin                h
    /bin/rm             xi
    /var                h
    /var/lib            h
    /var/lib/mydaemon   rwcdl
    ...
}
```

Listing 5: Ähnliche Einträge zusammenfassen

```
subject /bin/rm o {
        ...
        /mydir                  h
        /mydir/somefile         wd
        /mydir/someotherfile    wd
        /mydir/yetanotherfile   wd
        ...
    }

subject /bin/rm o {
    ...
    /mydir                  h
    /mydir/*                wd
    ...
}
```

Im Beispiel für *ping6* besitzt das Objekt / den Modus *h*, das heißt, a priori tappt das Programm in absoluter Dunkelheit. Ganz ohne Dateizugriffe geht es aber nicht. Dies beginnt mit den dynamisch gebundenen Laufzeitbibliotheken und endet bei verschiedenen, DNS-bezogenen Dateien. Im Listing sieht man gut die Optimierungen, die beim Ableiten der Policy stattfinden. *ping6* benötigt diverse Shared Librarys aus */lib*, aber nur eine einzelne aus */usr/lib*. Daher ist */lib* zunächst pauschal mit *rx* freigegeben, kritische Bereiche wie die Kernel-Module (*/lib/modules*) bleiben jedoch weiter unter Verschluss. Die einzelne Bibliothek aus */usr/lib* berücksichtigt dagegen ein direkter Eintrag.

Nach dem Festlegen der Objekte und der darauf zulässigen Zugriffe folgt eine Angabe der Capabilities, mit denen das Subjekt handeln darf. Listing 1 entzieht *ping6* zunächst alle Capabilities und gewährt anschließend *CAP_NET_RAW*. Das heißt, unabhängig davon, dass Root der Superuser ist, darf auch er *ping6* nur mit eingeschränkten Privilegien nutzen. Die letzten drei Einträge legen noch fest, dass *ping6* keine UDP- und TCP-Sockets nutzen darf – weder aktiv (*connect*) noch passiv (*bind*) – und dass auch IPv6 erlaubt ist.

Zum Abschluss einige Polierarbeiten

Weitere hier nicht gezeigte Möglichkeiten sind die Angabe subjektbezogener Ressourcenschranken sowie PaX-Flags. Ebenfalls nicht aufgeführt sind subjektspezifische Attribute, über die sich einschränken lässt, zu welcher UID oder GID ein Programm innerhalb des Subjekts wechseln darf, selbst wenn die entsprechenden Prozesse die Capabilities *CAP_SET_UID* beziehungsweise *CAP_SET_GID* besitzen. Dennoch kann man bereits gut erkennen, wie Grsecurity die Privilegien von Programmen auf das erforderliche Minimum zurechtstutzen kann.

Es empfiehlt sich, eine automatisch erzeugte Policy zunächst zu sichten und gegebenenfalls ein wenig nachzuarbeiten. Das soll zum einen unplausible Einträge in der produktiven Policy verhindern. So hat der Autor bei den Recherchen zum Artikel festgestellt, dass das System für den SSH-Dienst in der Benutzerrolle *root* zu jeder IP-Adresse das Rollenattribut *role_allow_ip* aufnimmt, von der aus ein Verbindungsversuch erfolgt, auch bei fehlgeschlagener Authentifizierung. Ist das System und im Speziellen der SSH-Dienst aus dem Internet nutzbar, landen so auch die IP-Adressen von Brute-Force-Angreifern im Regelwerk. Zum anderen gilt dasselbe Problem für die *connect*- und *bind*-Regeln eines Subjekts: So erzeugt der Trainingsmodus für einen Webserver für jede einzelne Adresse eine Regel. Für einen Webserver, der aus dem Internet aus erreichbar sein soll, lassen sich diese Regeln durch eine Regel wie *bind 0.0.0.0/0:80 stream tcp* zusammenfassen.

Daneben erstellt der Lernmodus von Grsecurity rollenbezogene Subjekte für jedes verwendete Kommando. Hier kommt eine weitere Stärke der Policy-Gestaltung von Grsecurity zum Tragen – Vererbung. Wie schon gesagt kann man Subjektregeln nicht nur für einzelne Programme, sondern auch für Verzeichnisse erstellen. Diese Regeln werden dann rekursiv an alle Programme im Baum unterhalb des Verzeichnisses vererbt.

Das in Listing 2 gezeigt Richtlinienfragment sagt, dass es für eine (hier nicht genannte) Rolle lesende und ausführende Zugriffe auf */bin*, */lib*, */usr/bin* und */usr/lib* gewährt und dass alle Prozesse über keine Capabilities außer *CAP_DAC_READ_SEARCH* verfügen. Das Programm */bin/foo* bekommt zusätzlich lesende und ausführende Rechte in */usr/libexec*, gleichzeitig wird ihm die Capability *CAP_DAC_READ_SEARCH* entzogen und *CAP_NET_ADMIN* gewährt. Zu beachten ist, dass die Ver-

erbung nicht für Socket-Regeln implementiert ist – man muss für jedes Subjekt jeweils eigene Regeln für *connect* und *bind* spezifizieren.

Soll die Vererbung für ein Subjekt nicht zum Tragen kommen, ist dies durch das Subjektmodus-Flag *o* (override) anzuzeigen. Dann sind die Regeln für alle Objekte, beginnend mit /, vollständig anzugeben; würde man im Beispiel beim Subjekt */bin/foo* den Modus *o* ohne weitere Änderung ergänzen, würde *gradm* die Policy mit der Begründung abweisen, dass vor der Definition des Objekts */usr/libexec* zunächst Definitionen für / und */usr* erfolgen müssen.

Beim Erstellen der Richtlinien aus dem Learning-Log erzeugt *gradm* für alle während der Lernphase genutzten Programme ein Subjekt mit dem Modus *o*. Für interaktiv via Kommandozeile aufgerufene Programme bieten sich geeignete Zusammenfassungen an, da Grsecurity sonst die Nutzungsmöglichkeiten dieser Kommandos unnatürlich einschränkt, wie das Listing 3 zeigt.

Steuern des Lernprozesses

In der entsprechenden Rolle darf man mit *rm* nur eine einzige Datei löschen. Das kann so beabsichtigt sein – ist es aber in der Regel nicht. Genauer gesagt, muss man zwischen den Fällen unterscheiden, in denen der Aufruf eines Kommando interaktiv oder von anderen Programmen wie *cron* erfolgt. In letzterem Fall kann eine derartige Einschränkung durchaus sinnvoll sein. Eine Möglichkeit, die für ein Subjekt spezifizierten Rechte an ein aufgerufenes Programm zu übertragen, bietet der Objektmodus *i* (inherit). Listing 4 zeigt für einen hypothetischen Daemon, wie der seine Zugriffsrechte auf */bin/rm* überträgt.

Ähnlich stellt sich die Situation im umgekehrten Fall dar: Eine Subjektdefinition wie im oberen Teil von Listing 5 wird man wahrscheinlich wie im unteren Teil zusammenfassen wollen.

Einige Zusammenfassungen lassen sich programmatisch über Einträge in der Datei */etc/grsec/learn_config* automatisieren. Eine Beschreibung findet sich im Wiki zu Grsecurity sowie in der mitgelieferten Datei *learn_config*. Einträge der Form

```
always-reduce-path    /mydir/always
```

legen beispielsweise fest, dass Grsecurity alle Zugriffe auf */mydir/always* oder auf Dateien in beliebigen Unterverzeichnissen darin auf */mydir/always* vereinigt. Es bildet statt jeweils einzelner Einträge in der erzeugten Policy nur einen Eintrag für */mydir/always*. Andere Spielarten sind

```
high-reduce-path     /mydir/high
dont-reduce-path     /mydir/dont
```

Hier findet eine Reduktion für */mydir/high* nur statt, wenn viele Zugriffe auf Dateien unterhalb des Verzeichnisses erfolgen, während keine Reduktion für */mydir/dont* erfolgt. So lässt sich eine sehr differenzierte Policy erzeugen.

Für Anwendungen, die weitere Programme starten, lässt sich die Rechtevererbung steuern. Ruft beispielsweise */bin/foo* verschiedene andere Programme auf, die in diesem Kontext dieselben Zugriffsrechte wie */bin/foo* erhalten sollen, kann man dies so ausdrücken

```
inherit-learn      /bin/foo
```

Ein weiteres Steuern des Lernprozesses erfolgt über die Angabe besonders schutzbedürftiger Objekte wie */etc/ssh*, */etc/ssl*, */etc/shadow* oder */lib/modules*. Die Angabe

```
protected-path        /sys
high-protected-path   /etc/ssh
```

bewirkt, dass Grsecurity für jedes Programm, das auf diese Dateien zugreift, ein eigenes Subjekt anlegt. Die als „high protected" gekennzeichneten Objekte erhalten zudem stets das Objektmodus-Flag *h* und sind damit allgemein unsichtbar.

Sind Änderungen an der Policy vorzunehmen, muss man nicht gleich eine vollständig neue Lernphase durchlaufen. Dabei spielt es keine Rolle, ob sich das Verhaltensprofil geändert hat, eine Software nach einem Upgrade eine andere Dateistruktur aufweist, neue Rollen angelegt wurden oder man ein bestehendes Profil verfeinern möchte. Grsecurity bietet an, für spezifische Rollen oder Subjekte den Lernmodus zu aktivieren, während die Policy für alles andere scharf geschaltet bleibt. Hierzu fügt man der Rolle oder dem Subjekt das Modus-Flag *l* (learn) hinzu, beispielsweise *role pkiadmin sGl* oder *subject /usr/sbin/sshd l*.

Schutz vor Informationsabfluss

Das Aktivieren und Deaktivieren von Grsecurity erfolgt über *gradm -E -L /etc/gradm/learn.log* beziehungsweise über *gradm -D*; das in die Policy einzubauende Fragment erzeugt das Kommando *gradm -L /etc/gradm/learn.log -O /etc/grsec/newpolicy*. Darüber hinaus bietet Grsecurity einige spannende Funktionen zur weiteren Härtung, die der Lernmodus nicht aktiviert. So lassen sich vor allem in der Subjektdefinition Modus-Flags angeben, die der fortgeschrittenen Härtung dienen. Diese Flags stellt der nächste Abschnitt vor.

Normalerweise sind verschiedene Attribute eines Prozesses über per */proc/<pid>/* erreichbare Pseudodateien allgemein einsehbar – Tools wie *ps* oder *top* nutzen diese intensiv. Grsecurity schränkt die Sichtbarkeit von */proc* ohnehin stark ein, sodass ein Prozess, der nicht einer speziellen Gruppe angehört (meist GID 10, „wheel") nur Informationen über Prozesse mit derselben UID einsehen kann. Möchte man noch innerhalb einer UID zwischen verschiedenen Prozessen unterscheiden, muss man das Subjektmodus-Flag *h* (hide) für Programme verwenden, deren Prozessinformationen unsichtbar sein sollen. Diese lassen sich dann nur von Prozessen einsehen, deren Modus das Flag *v* (view) enthält.

Daneben lassen sich verschiedene Informationen über Prozesse zu schützen, ohne diese Prozesse zu verbergen. Das Subjektmodus-Flag *d* verhindert den Zugriff auf die File-Deskriptoren, die Speicherbelegung, die vollständige Kommandozeile (einschließlich aller Parameter) sowie die Umgebungsvariablen. Ein aufhebendes Flag analog zu *v* existiert jedoch nicht. Dieser Modus bietet sich vor allem für Programme an, die vertrauliche Informationen nutzen, zum Beispiel Passwörter oder Schlüssel.

In eine ähnliche Richtung zielt das Subjektmodus-Flag *A* zum Schutz von Shared Memory. Dieser Mechanismus kommt oft zur effizienten Interprozesskommunikation zum Einsatz. Sind die

Listing 6: *include*, *replace* und *define*

```
/etc/grsec/policy:

    include </etc/grsec/defns>
    ...

    subject /bin/foo {
  /etc    r
  $etc_denied
  $FOOLIB   r
  ...
    }

    subject /bin/bar {
  /etc    r
  $etc_denied
  /usr/lib  rw
  $FOOLIB   h
  ...
    }
    ...

/etc/grsec/defns:

    replace FOOLIB  /usr/lib/foo

    define etc_denied {
       /etc/ssh   h
       /etc/ssl   h
       /etc/shadow  h
       /etc/shadow- h
       ...
    }
```

Parameter für einen Shared-Memory-Abschnitt bekannt, kann ein Prozess derselben UID darauf zugreifen. Das Flag *A* bewirkt, dass ausschließlich Prozesse (das heißt Objekte) in diesem Subjekt auf das Shared Memory des Subjekts zugreifen können. Das Anwendungsszenario ähnelt dem von *d*.

Schutz der Systemintegrität

Für den Umgang mit unbotmäßigen Programmen dient das Subjektmodus-Flag *K*. Es legt fest, dass das System einen unter den Objekten eines Subjekts aufgeführten Prozess terminiert, wenn er wegen einer Regelverletzung eine Grsecurity-Warnung hervorruft. Noch brachialer geht Grsecurity beim Subjektmodus-Flag *C* vor: Die Software führt für jeden Prozess die IP-Adresse mit, auf die sich dessen Ausführung zurückführen lässt. Beim Auftreten einer Grsecurity-Warnung beendet Grsecurity sämtliche mit dieser IP-Adresse assoziierten Prozesse – unabhängig davon, ob sie zur Regelverletzung beigetragen haben oder nicht. Grsecurity nutzt das Konzept für ausführbare, aber nicht schreibbare Programmdateien in Form von Trusted Path Execution (TPE). TPE lässt sich auf verschiedene Arten nutzen, dieser Artikel beschränkt sich auf eine Beschreibung der Nutzung im Kontext von RBAC und MAC. Hierfür dient das Flag *T*.

Als Rollenmodus-Flag führt *T* dazu, dass Nutzer in der Rolle keine schreibbaren Programme ausführen können. Die Prüfung erfolgt dabei zur Laufzeit. Als Subjektmodus-Flag bewirkt *T*, dass Grsecurity RBAC nur dann aktivieren kann, wenn die darunter gefassten Programme und Skripte durch ein beliebiges anderes Subjekt schreibbar sind; ausgenommen hiervon sind Subjekte in administrativen Rollen. Die Prüfung erfolgt zum Zeitpunkt der Aktivierung von Grsecurity RBAC.

Ein weiteres Subjektmodus-Flag dient dem Schutz vor trojanischen Shared Libraries: *s*. Es aktiviert einen Mechanismus, der auch beim Aufruf von SUID-/SGID-Programmen zum Tragen kommt. Der setzt das Prozessattribut *AT_SECURE* mit der Folge, dass unter anderem der Linker (*/lib/ld-linux.so*) die sicherheitskritischen Umgebungsvariablen *LD_LIBRARY_PATH, LD_PRELOAD, LD_AUDIT* et cetera ignoriert.

Strukturieren der Policy

Unter */etc/grsec/policy* findet sich die von Grsecurity genutzte Policy. Bei vielen Rollen, Benutzern und Subjects kann die Datei jedoch recht unübersichtlich werden. Daher darf man die Policy in mehrere Dateien aufteilen, die die Direktive *include* in die Hauptdatei einbindet. So bietet es sich an, die Definition der einzelnen Rollen in separate Dateien auszulagern.

Daneben bietet Grsecurity die Option, über die Direktive *replace* Platzhalter zu definieren, sowie über *define* oft verwendete Blöcke zu benennen. Listing 6 zeigt ein Beispiel für den Einsatz der drei genannten Direktiven.

Über diese Direktiven lässt sich selbst eine komplexe Policy noch übersichtlich gestalten. Wer noch raffiniertere Möglichkeiten zur Gestaltung braucht, etwa die Nutzung von Templates für verschiedene Rollen, der sollte auf Makroprozessoren wie *m4* zurückgreifen.

Fazit

Grsecurity weist viele gute Ansätze zur Systemhärtung auf, von denen der Artikel nur eine Auswahl anreißen konnte. Als dickster Minuspunkt ist zu vermerken, dass Grsecurity nicht Bestandteil des offiziellen Kernels ist. Dies bedeutet beispielsweise, dass IT-Betreiber, die aus Support-Gründen ein Linux „von der Stange“ nehmen müssen, nur unter erschwerten Bedingungen in den Genuss von Grsecurity kommen – wenn überhaupt. Ein weiterer kritischer Punkt besteht in der Abhängigkeit von Brad Spengler, dem Hauptentwickler von Grsecurity. Die Zukunft des Projekts stand schon mehrmals auf der Kippe, aber zumindest derzeit gibt es genügend Sponsoren, die über ihre Unterstützung die Pflege und Weiterentwicklung von Grsecurity ermöglichen.

Darüber hinaus lässt die verfügbare Dokumentation oft noch an Klarheit zu wünschen übrig. Obwohl Grsecurity eigentlich relativ leicht zu bedienen ist, erfordert die Klärung von Details zur Wirkung verschiedener Mechanismen bisweilen einige Experimente.

Aus Sicht der Policy wäre es wünschenswert, automatische Transitionen in spezielle Rollen zu ermöglichen; dies ist beispielsweise unter SELinux möglich. Ideal wäre es, wenn man eine derartige Transition über eine Spezifikation verschiedener Programme veranlassen könnte. Im bestehenden Regelwerk lässt sich dies nicht ausdrücken – Brad Spengler hat dazu auch durchblicken lassen, dass er darin keinen Gewinn sieht. Ein weiteres Manko besteht darin, dass ein Prozess in der Regel nicht feststellen kann, in welcher Rolle er sich gerade befindet.

Mittelfristig wäre es auch erforderlich, IPv6-Adressierung zu integrieren, derzeit lässt sich lediglich die komplette Nutzung gewähren oder verweigern. Zu wünschen wäre es ebenfalls, den Vererbungsmechanismus (zumindest optional) für Subjekte auf Socket-Regeln auszudehnen.

Doch trotz aller Kritik kommt man nicht an der Feststellung vorbei, dass Grsecurity ein echtes Highlight der IT-Sicherheit ist und dass die vom Team um Brad Spengler geschaffene Software Hand und Fuß hat. Auf dem Weg zu einer echten Härtung für den Einsatz in potenziell feindseligen Umgebungen jenseits des schablonenhaften Abarbeitens üblicher Härtungs-Guides stellt Grsecurity zumindest einen großen Sprung nach vorn dar. (avr)

Onlinequellen

T[a]	Ebury-Rootkit	www.welivesecurity.com/wp-content/uploads/2014/03/operation_windigo.pdf
[b]	Center for Internet Security	benchmarks.cisecurity.org/downloads/
[c]	J. Edge: The future for Grsecurity	lwn.net/Articles/313621/
[d]	M. Fox et al.: SELinux and grsecurity	www.cs.virginia.edu/~jcg8f/GrsecuritySELinuxCaseStudy.pdf
[e]	NixCraft: Linux Kernel Security	www.cyberciti.biz/tips/selinux-vs-apparmor-vs-grsecurity.html
[f]	Funktionsübersicht	grsecurity.net/features.php
[g]	B.Spengler: RBAC-Tutorial	grsecurity.net/rbac_tutorial.pdf
[h]	Grsecurity Wikibook	en.wikibooks.org/wiki/Grsecurity

Dr. Safuat Hamdy
arbeitet als Security Consultant bei der Secorvo Security Consulting GmbH in Karlsruhe.

Alle Links: www.ix.de/ix1416008

Systemüberprüfungen mit OpenSCAP

Verteilte Absicherung

Thorsten Scherf

Für immer mehr Firmen ist Compliance-Management ein wichtiger Aspekt bei der Einrichtung, Wartung und Beschreibung ihrer IT-Landschaft. In der Praxis gestaltet sich die Umsetzung jedoch nicht immer einfach. Hier verspricht SCAP Abhilfe, das Security Content Automation Protocol. Damit und mit den Tools aus dem OpenSCAP-Paket lässt sich die Einhaltung von IT-Richtlinien komfortabel überprüfen.

Das Thema Compliance-Management umfasst viele verschiedene Themengebiete und lässt sich demzufolge unter unterschiedlichsten Aspekten betrachten. Vereinfacht gesprochen fasst man unter dem Begriff alle in einem Unternehmen eingerichteten Maßnahmen und Prozesse zusammen, die die Einhaltung geltender Gesetze und Richtlinien sicherstellen sollen. In diesem Umfeld existiert eine Reihe international anerkannter Zertifizierungen, besonders wichtig sind hier die Evaluation Assurance Level der Common Criteria (CC EAL, siehe „Onlinequellen“, [a]) sowie die ISO-27001-Zertifizierung [b]. In Deutschland kann das Bundesamt für Sicherheit in der Informationstechnik (BSI) beide vergeben, wenn die zu evaluierende IT-Landschaft den Anforderungen entspricht. Für ISO 27001 muss sie konform zu den Anforderungen des BSI-IT-Grundschutzes sein, für die EAL-Zertifizierung die Anforderungen der einzelnen Common-Criteria-Level erfüllen.

Dabei setzen die meisten Zertifizierungen bestimmte Systemkonfigurationen voraus. Diese wiederum basieren auf Richtlinien sowie diversen Konfigurationsbeschreibungen für ein IT-System, die – wie bei EAL – teilweise eine bestimmte Hardwarekonfiguration umfassen können. Bei ISO 27001 kommen weitere Gesichtspunkte hinzu, beispielsweise sind hier auch ITIL-relevante Themen auf dem Prüfstand. Die Systemkonfiguration bildet hier also nur einen kleinen Teilbereich der Zertifizierungsanforderungen.

Dieser Artikel behandelt primär die zu erfüllenden Richtlinien zur Systemkonfiguration, strebt man eine entsprechende Zertifizierung an oder möchte man die eigenen Systeme einfach unter

Best-Practice-Gesichtspunkten konfigurieren. Von diesen Vorgaben existiert eine ganze Reihe. Teilweise überschneiden sich diese sogar oder sprechen Empfehlungen aus, die in einer anderen Richtlinie komplett fehlen. Hier muss man vor dem Hintergrund der angestrebten Zertifizierung darauf achten, welche Richtlinie für die eigene Systemlandschaft am ehesten passt.

Unterschiedliche Richtlinien

So sind beispielsweise für sämtliche Systeme des US-Verteidigungsministeriums die Secure Technical Implementation Guides (STIG) [c] der DISA FSO (Defense Information Systems Agency, Field Security Operations) maßgebend. Der Standard für Systeme in der Kreditkartenindustrie ist hingegen PCI-DSS, der Payment Card Industry Data Security Standard [d]. Daneben gibt es noch allgemeinere Härtungsrichtlinien wie die Security Configuration Benchmarks vom Center for Information Security (CIS) [e]. Alle diese Richtlinien beschränken sich primär auf die Beschreibung, welche Einstellungen auf einem IT-System vorzunehmen sind, damit es als sicher gilt und somit dem jeweiligen Standard entspricht. Aspekte, die unter dem allgemeinen Begriff des IT-Service-Managements fallen, oder auch globale Fragen der physischen Gebäudesicherheit sind nicht Gegenstand solcher Härtungsmaßnahmen.

Im Folgenden geht es vor allem um die STIG der DISA FSO und die CIS Security Configuration Benchmarks, da es sich bei diesen um den De-facto-Standard bei allgemeinen Härtungsrichtlinien für Linux-Systeme handelt. Alle hier getätigten Aussagen lassen sich aber genauso auch auf andere Richtlinien anwenden. Beide Organisationen stellen sogenannte Security Re-

In SCAP berücksichtigte Standards

CCE	Common Configuration Enumeration
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
OVAL	Open Vulnerability and Assessment Language
XCCDF	Extensible Configuration Checklist Description Format
ab Version 1.2 zusätzlich	
ARF	Asset Reporting Format
CCSS	Common Configuration Scoring System
OCIL	Open Checklist Interactive Language
TMSAD	Trust Model for Security Automation Data

In OVAL-Repository finden sich aktuell mehr als 22 000 Definitionen; ein Großteil davon entfällt auf Windows und Unix/Linux (Abb. 1).