Paketmanagement auf Oracle Linux mit RPM und yum

Lenz Grimmer
Oracle Deutschland B.V. & Co. KG
Hamburg, Germany

Schlüsselworte

Oracle Linux, Software-Management, RPM, yum, Pakete, Administration, Updates, Sicherheit

Einleitung

Oracle Linux verwendet, wie ein Großteil der bekannteren Linux-Distributionen, den RPM-Paketmanager zur Installation und Verwaltung von Software-Paketen. RPM bietet eine Reihe von nützlichen Funktionen, die die Installation, Aktualisierung und das Entfernen einzelner Softwarepakete sehr erleichtern. Die Verwendung des `yum` Paketmanagers bietet auf RPM aufbauende weitergehende Funktionalität wie die automatische Auflösung von Paketabhängigkeiten und -konflikten.

Dieser Vortrag gibt eine Einführung in die Grundfunktionalität von RPM und yum und gibt Einblicke, wie die alltägliche Arbeit eines Systemadministrators mit diesen Werkzeugen erleichtert und unterstützt werden kann. Konkrete Kommandozeilen-Beispiele unterstützen die Vermittlung von Tricks und Kniffen, mit denen sich häufig vorkommende Problemstellungen lösen lassen.

RPM – der RPM Package Manager

Zur Verteilung und Installation von Softwarepaketen auf Linux-Distributionen haben sich zwei verschiedenen Formate als de-facto Standards etabliert. Einerseits gibt es das auf Debian und dessen Derivaten (wie z.B. Linux Mint oder Ubuntu Linux) verwendete DEB-Format, auf der anderen Seite das ursprünglich von Red Hat entwickelte RPM-Format.

Bekannte RPM-basierte Distributionen sind neben Red Hat Enterprise Linux (und dessen Abkömmlingen wie Oracle Linux, CentOS oder Scientific Linux) beispielsweise Fedora Linux, Mageia oder SUSE Linux. Aufgrund dieser weiten Verbreitung wurde der ursprüngliche Name “Red Hat Package Manager” später in das rekursive Akronym “RPM Package Manager” geändert. Tatsächlich wird RPM auch auf anderen Plattformen verwendet – so verwendet zum Beispiel IBM's „AIX Toolbox for Linux Applications“ das RPM-Paketformat, um Software auf dem AIX-Betriebssystem zu verwalten.

RPM-Pakete bestehen im Prinzip aus einer mit `gzip` oder `bzip2` komprimierten CPIO-Archivdatei, die mit einem zusätzlichen Header versehen wird. CPIO ist ähnlich wie `tar` sowohl ein Archivierungsprogramm als auch ein Dateiformat, mit dem Dateien sequentiell in eine Archivdatei oder auf ein Bandlaufwerk geschrieben werden können. Im RPM-Header werden die zum Paket gehörigen Meta-Informationen wie z.B. Paketname, Beschreibung, Version, Lizenz und Erstellungsdatum, sowie Abhängigkeiten von anderen Paketen, kryptographische Signaturen und Checksummen gespeichert. Binäre RPM-Pakete enthalten weiterhin Plattform-spezifische Informationen, wie z.B. die unterstützte Hardware-Architektur. Mit Hilfe der Signaturen und Checksummen kann überprüft werden, dass ein Paket intakt ist und nicht nach der Freigabe durch den Ersteller modifiziert wurde.

RPM ist sowohl der Name des Paketformats als auch des für die Verwaltung der Pakete zu verwendenden Kommandozeilen-Werkzeugs. Dieses erfüllt zwei grundlegende Funktionen.

Zum einen ist es ein Werkzeug zur automatisierten Erstellung von binären, installierbaren Softwarepaketen aus dem Quelltext einer Anwendung. Sämtliche zum Bau eines RPM-Pakets erforderlichen Schritte sind in einem Script, der sogenannten "Spec-Datei" vermerkt und können damit jederzeit wiederholt werden. Sie umfassen üblicherweise den klassischen "Open Source Dreisprung" aus den Schritten `configure`, `make` und `make install`, mit dem der Quelltext eines Programms in installierbare Binärdateien überführt wird. Darüber hinaus enthält die Spec-Datei alle Informationen, die später im Paketheader abrufbar sind.

RPM erzeugt aus den Eingabedateien (Source-Tar-Archiv, evtl. Patches, Spec-Datei) im Build-Prozess ein Quell-Archiv (das Source-RPM oder SRPM), sowie ein oder mehrere Binärpakete (RPMs).

Diese daraus resultierenden RPM-Pakete können dann wiederum mit Hilfe von RPM auf einem System installiert, aktualisiert oder auch rückstandsfrei wieder entfernt werden. RPM kann Pakete installieren, die lokal im Dateisystem abgelegt sind, oder sie vor der Installation mit Hilfe des eingebauten http/ftp-Clients von einem anderen Server herunterladen.

RPM führt Buch über alle auf dem System installierten Software-Pakete. Diese auf BerkeleyDB basierte Datenbank verwaltet zu jedem Paket Dateilisten, Checksummen und Zeitstempel der installierten Dateien sowie Dateitypen.

RPM behandelt Konfigurationsdateien mit besonderer Vorsicht, um manuelle Anpassungen bei einem späteren Paket-Update nicht zu überschreiben. Je nach Applikation kann der Paketentwickler bestimmen, ob die veränderte Konfigurationdatei nach dem Update weiter verwendet werden soll, oder als Sicherheitskopie beiseite geschoben wird (wenn sich z.B. das Format der Konfigurationsdatei durch den Update verändert). In diesem Fall kann der Systemadministrator eventuelle Änderungen manuell von der Sicherheitskopie übertragen.

Darüber hinaus verwaltet RPM Abhängigkeiten zwischen Paketen. Dies stellt sicher, dass einerseits beim Entfernen von Paketen keine Dateien (wie z.B. System-Bibliotheken) entfernt werden, die noch von anderen auf dem System installierten Applikationen benötigt werden.

Die Erstellung der Abhängigkeiten zu Laufzeitbibliotheken (shared libraries) oder Skriptsprachen wie Python oder Perl erfolgt durch eine automatische Analyse des Paketinhalts während des Build-Prozesses. Weiterhin kann der Paket-Ersteller manuell weitere Abhängigkeiten hinzufügen, die ebenfalls in den Metadaten des RPM-Headers vermerkt werden.

Bei der Installation eines RPM-Pakets wird vorab überprüft, ob alle Abhängigkeiten erfüllt sind und keine Konflikte zu anderen bereits installierten Paketen bestehen. Falls dies nicht der Fall ist, bricht RPM die Installation mit einer entsprechenden Meldung ab und der Anwender muss zuerst manuell die fehlenden Pakete nachinstallieren. Dieser Umstand hat RPM (zu Unrecht) einen schlechten Ruf eingebracht und wird mitunter auch als "dependency hell" bezeichnet, da die Abhängigkeitsketten mitunter sehr lang werden können.

Insbesondere der Umstand, die von RPM monierten Paketabhängigkeiten und -konflikte manuell auflösen zu müssen, hat einige Entwickler auf den Plan gerufen, diesen Vorgang zu automatisieren. Im Laufe der Zeit entstanden diverse Werkzeuge und Frontends wie `apt-rpm`, SuSE's `zypper`, Red Hat's `up2date` oder `yum`, die die Funktionalität von RPM um weitere Fähigkeiten erweiterten.

Im Folgenden soll yum, der “Yellowdog Updater, Modified” weiter behandelt werden.

yum — das RPM-basierte Paketmanagement-System

yum, der “Yellowdog Updater, Modified” wurde urspünglich von Seth Vidal für die auf der PowerPC-Architektur sehr beliebte Yellowdog Linux-Distribution entwickelt und wurde später über Fedora Linux auch auf Red Hat Enterprise Linux (und damit auch dessen Derivaten wie z.B. Oracle Linux) übernommen.

Yum ist wie RPM ein Kommandozeilen-Programm, allerdings sind ebenfalls diverse grafische Frontends (wie z.B. PackageKit) dafür verfügbar.

Yum ermöglicht die Suche, Installation, Aktualisierung und Entfernung von RPM-Paketen auf einem Linux-System. Eventuelle Abhängigkeiten zu anderen Paketen werden dabei berücksichtigt und nach Möglichkeit automatisch aufgelöst. Dies bedeutet, dass die Installation eines bestimmten Pakets mitunter eine ganze Reihe weiterer Pakete nach sich zieht, z.B. wenn eine Applikation weitere Systembibliotheken benötigt, die zuvor noch nicht installiert waren.

Weiterhin erweitert yum RPM um das Konzept der Paket-Repositories und Gruppen. RPM agiert typischerweise auf einzelnen RPM-Paketen, während ein yum-Repository eine nahezu beliebige Anzahl von RPM-Paketen enthalten kann. Diese Repositories können dabei sowohl auf lokalen Verzeichnissen, Installationsmedien wie CDs/DVDs, HTTP- oder auch FTP-Servern abgelegt sein und enthalten neben den eigentlichen RPM-Paketen weitere Dateien mit Metadaten im XML-Format, die den Inhalt des Repositories näher beschreiben. Weiterhin können einzelne RPMs zu Gruppen zusammengefasst werden, die dann auf einen Schlag zusammen mit Hilfe von yum installiert werden können.

Wird yum aufgerufen, so durchsucht es zuerst alle eingetragenen Repositories nach neuen oder aktualisierten Paketen. Danach lädt es die für eine Installation oder Update erforderlichen RPM-Pakete von den entsprechenden Repositories herunter und legt sie temporär im Dateisystem ab. Für den eigentlichen Installationsvorgang und die Verwaltung der Paketdatenbank ruft yum anschließend RPM im Hintergrund auf.

Yum bietet durch seine Plugin-Schnittstelle nahezu beliebige Erweiterungsmöglichkeiten. Plugins werden (wie yum selbst) in der Skriptsprache Python entwickelt und erweitern die Grundfunktionalität in vielen Bereichen.

Zu den Funktionen, die bereits durch Plugins realisiert werden, gehören unter anderem die Überwachung der Verbindungsgeschwindigkeiten zu den Repository-Servern und die Auswahl des jeweils schnellsten Servers (Plugin `yum-plugin-fastestmirror`). Diese dynamische Konfiguration ermöglicht auch das Ausweichen auf andere Server im laufenden Betrieb, falls ein Server ausfällt oder sich als nicht aktuell herausstellen sollte. Dies erhöht die Verfügbarkeit und Robustheit der Update-Infrastruktur.

Das “Yum Security Plugin” (`yum-plugin-security`) dient zur Verwaltung und Installation von Updates für RPM-Pakete mit bekannten Sicherheitslücken. Filter ermöglichen die gezielte Suche nach bestimmten Sicherheitslücken anhand der CVE-Kennungen, oder der Dringlichkeit (Severity) eines Sicherheitsproblems. Ein Anwendungsbeispiel dazu befindet sich im nachfolgenden Kapitel.

Das “Yum Filesystem Snapshot Plugin” (`yum-plugin-fs-snapshot`) erstellt mit Hilfe des Linux Volume Managers (LVM) oder des Btrfs-Dateisystems Schnappschüsse eines installierten Sys-

tems vor Veränderungen und erlaubt damit im Notfall einen “Rollback” auf den vorherigen Zustand des Systems.

Kommandozeilen-Beispiele

Im Folgenden sollen einige häufig in der Praxis vorkommende Aufgabenstellungen anhand konkreter Beispiele vorgestellt werden.

Ansicht des Pakethistorie

Jedes RPM-Paket enthält ein Protokoll, in dem der Paketentwickler relevante Veränderungen und Verbesserungen dokumentiert. Hierbei handelt es sich meist um Veränderungen am RPM-Paket und der Paketierung der Applikation an sich, die detaillierte Entwicklungshistorie der jeweiligen Applikation ist üblicherweise in einer eigenen, ebenfalls im RPM-Paket enthaltenen Changelog-Datei festgehalten.

Die im RPM festgehaltenen Änderungskommentare enthalten einen Zeitstempel und den Namen und die E-Mail-Adresse des Entwicklers, der die Änderungen vorgenommen hat.

Das folgende Beispiel zeigt die letzten Änderungen im RPM des Unbreakable Enterprise Kernel (UEK) auf Oracle Linux:

```
[oracle@oraclelinux6 ~]$ rpm -q --changelog kernel-uek | head -20

* Fri Feb 22 2013 Guru Anbalagane <guru.anbalagane@oracle.com> [2.6.39-
400.17.1.el6uek]

- This is a fix on dlm_clean_master_list() (Xiaowei.Hu)

- RDS: fix rds-ping spinlock recursion (jeff.liu) [Orabug: 16223050]

- vhost: fix length for cross region descriptor (Michael S. Tsirkin)
[Orabug:

  16387183] {CVE-2013-0311}

- kabifix: block/scsi: Allow request and error handling timeouts to be

  specified (Maxim Uvarov)

- block/scsi: Allow request and error handling timeouts to be specified
(Martin

  K. Petersen) [Orabug: 16372401]

- [SCSI] Shorten the path length of scsi_cmd_to_driver() (Li Zhong)
[Orabug:

  16372401]

- Fix NULL dereferences in scsi_cmd_to_driver (Mark Rustad) [Orabug:
16372401]

- SCSI: Fix error handling when no ULD is attached (Martin K. Petersen)

  [Orabug: 16372401]

- Handle disk devices which can not process medium access commands (Martin
```

```
K.

  Petersen) [Orabug: 16372401]

- the ac->ac_allow_chain_relink=0 won't disable group relink (Xiaowei.Hu)

  [Orabug: 14842737]

- pci: hotplug: fix null dereference in pci_set_payload() (Jerry
Snitselaar)

  [Orabug: 16345420]
```

Abfrage der RPM-Datenbank: Zu welchem RPM-Paket gehört diese Datei?

Die RPM-Datenbank des Linux-Systems führt nicht nur darüber Buch, welche Softwarepakete installiert sind, sondern auch, welche Dateien zu einem Paket gehören. Auf diese Weise ist es leicht möglich herauszufinden, zu welchem Paket eine bestimmte Datei gehört:

```
[oracle@oraclelinux6 ~]$ rpm -qf /boot/vmlinuz-2.6.39-
400.17.1.el6uek.x86_64

kernel-uek-2.6.39-400.17.1.el6uek.x86_64
```

Allerdings ist nicht jede im Dateisystem installierte Datei in der RPM-Datenbank verzeichnet. Dazu gehören z.B. Dateien die dynamisch zur Laufzeit (z.B. durch Skripte) erzeugt werden, oder deren Dateinamen veränderlich sind (z.B. Log-Dateien mit Zeitstempeln oder die zu einer Applikation gehörenden Arbeitsdaten):

```
[oracle@oraclelinux6 ~]$ rpm -qf /etc/aliases /etc/aliases.db

setup-2.8.14-20.el6.noarch
file /etc/aliases.db is not owned by any package
```

In diesem Fall ist die Text-Eingabedatei `/etc/aliases` dem Paket `setup` zugeordnet, während die daraus durch das `newaliases`-Kommando erzeugte binärdatei `/etc/aliases.db` nicht in der Datenbank erfasst ist.

Auflistung der zu einem Paket gehörigen Dokumentation und Konfigurationsdateien

RPM unterscheidet verschiedene Dateitypen, die in einem Paket enthalten sein können: reguläre Dateien (wie z.B. ausführbare Binaries, Bibliotheken), Konfigurationsdateien (üblicherweise Text-Dateien, die in der Verzeichnisstruktur unter `/etc` abgelegt sind) oder Dokumentation (wie z.B. manual pages oder README-Dateien). Es ist möglich, die RPM-Datenbank explizit nach diesen Attributen zu befragen, um z.B. sämtliche zu einem Paket gehörige Dokumentation aufzulisten:

```
[oracle@oraclelinux6 ~]$ rpm -qd psutils

/usr/share/doc/psutils-1.17/LICENSE
/usr/share/doc/psutils-1.17/README
/usr/share/man/man1/epsffit.1.gz
/usr/share/man/man1/getafm.1.gz
/usr/share/man/man1/psbook.1.gz
/usr/share/man/man1/psnup.1.gz
/usr/share/man/man1/psresize.1.gz
/usr/share/man/man1/psselect.1.gz
```

```
/usr/share/man/man1/pstops.1.gz
```

Analog dazu listet das folgende Kommando alle zum Postfix-Paket gehörenden Konfigurationsdateien auf:

```
[oracle@oraclelinux6 ~]$ rpm -qc postfix

/etc/pam.d/smtp.postfix
/etc/postfix/access
/etc/postfix/canonical
/etc/postfix/generic
/etc/postfix/header_checks
/etc/postfix/main.cf
/etc/postfix/master.cf
/etc/postfix/relocated
/etc/postfix/transport
/etc/postfix/virtual
/etc/sasl2/smtpd.conf
```

Überprüfung der Paket-Integrität

RPM kann auch nützlich sein, um die Integrität aller einem Paket zugeordneten Dateien zu überprüfen. RPM speichert für jede installierte Datei eine MD5-Prüfsumme in seiner Datenbank. Diese Funktionalität kann dazu verwendet werden, um die Integrität der installierten Pakete auf einem System gegen diese Signatur-Datenbank zu überprüfen (ähnlich dem Werkzeug „Tripwire“). Neben MD5-Prüfsummen werden noch weitere Eigenschaften einer Datei wie z.B. die Größe, Zugriffsrechte und das Modifikationsdatum in der RPM-Datenbank abgespeichert. Diese Informationen können nützlich sein, um eine schnelle Beurteilung durchzuführen, ob ein System durch einen Hacker-Angriff kompromittiert wurde und ob Dateien (z.B. ausführbare Dateien) modifiziert oder durch Trojaner-Versionen ersetzt wurden.

Beachten Sie jedoch, dass ein cleverer Hacker neben den Dateien auch die RPM-Datenbank selbst modifiziert haben könnte, so dass Sie immer noch zusätzliche Kontrollen durchführen sollten, wenn Sie Grund zu der Annahme haben, dass Ihr System kompromittiert wurde, vorzugsweise durch das Booten des Systems von einem nichtmodifizierbaren Boot-Image (wie z.B. einer Live-CD).

Die Integrität eines einzelnen installierten Software-Pakets läßt sich mit dem Kommando `rpm -V <Paketname>` überprüfen.

Die Überprüfung aller Dateien mit `rpm -Va` kann eine Weile dauern und erzeugt eine Menge an Information, die auf Korrektheit überprüft werden muß.

Es ist möglich, die Verifizierung mit Hilfe der Optionen `--nogroup`, `--nouser` oder `--nomtime` auf bestimmte Kriterien einzuschränken.

Beispiel:

```
[oracle@oraclelinux6 ~]$ rpm -V initscripts

..5....T.  c /etc/inittab
```

Das aus 8 Zeichen bestehende Feld vor dem Dateinamen zeigt auf, was genau sich im Vergleich zur in der RPM-Datenbank gespeicherten Information geändert hat. Der einzelne Buchstabe weist auf mögliche Datei-Eigenschaften hin, wie z.B. “c” für eine Konfigurationsdatei oder “d” für Dokumentation.

Jedes der 8 Zeichen zeigt das Ergebnis eines Vergleichs der Datei mit den in der RPM-Datenbank gespeicherten Attributen an. Ein einzelner Punkt (“.”) bedeutet, dass der Test erfolgreich war, ein einzelnes Fragezeichen (“?”) deutet auf eine nicht durchgeführte Prüfung hin (z.B. wenn die Zugriffsrechte der Datei eine Auswertung nicht zulassen). Ansonsten zeigt ein entsprechender Buchstabe eine Differenz zu den in der Datenbank gespeicherten Informationen auf:

- S – Dateigröße stimmt nicht überein
- M – Dateimodus verändert (beinhaltet Zugriffsrechte und Dateityp)
- 5 – MD5 Checksumme weicht ab
- D – Major/minor Gerätenummer stimmt nicht überein
- L – Diskrepanz im Linkpfad
- U – Benutzerzugehörigkeit unterschiedlich
- G – Gruppenzugehörigkeit unterschiedlich
- T – Zeitstempel der letzten Veränderung
- P – Dateifunktionen („capabilities“) unterscheiden sich

Im obigen Beispiel wurde `/etc/inittab` verändert, als der Standard-Runlevel durch den Installer während der Erstinstallation des Systems von 3 (Textmodus) nach 5 (GUI-Modus) gewechselt wurde.

Dies resultierte in einer Checksummen-Differenz und einem abweichenden Zeitstempel – was zu erwarten war, nachdem die Datei editiert wurde.

Abfrage und Installation sicherheitsrelevanter Paket-Updates

Um ein Linux-System stets auf dem aktuellen Stand zu halten und gegen neu bekannt gewordene Sicherheitslücken zu schützen, empfiehlt es sich, Paketaktualisierungen möglichst zeitnah zu installieren. Die Verwaltung von Sicherheits-Updates wird mit Hilfe des Yum Security Plugin vereinfacht. Mit Hilfe des Plugins und spezieller vom Repository bereitgestellter Informationen ist es möglich, die Installation von Updates auf bestimmte Kriterien zu beschränken. Ein Beispiel wären sicherheitsrelevante Aktualisierungen.

Das Kommando `yum updateinfo security` (oder `yum list-security`) gibt eine Liste sämtlicher anstehender Sicherheits-Updates und der dazugehörigen CVE-Nummer aus. Das CVE-System (Common Vulnerabilities and Exposures) ist ein Industriestandard zur Benennung und Identifizierung von Sicherheitslücken in Computersystemen und wird von der US-amerikanischen MITRE Corporation verwaltet.

Oracle Linux unterstützt dieses System, indem Paket-Updates mit diesen CVE-Nummern versehen werden. Diese Information ist öffentlich einsehbar – unter http://linux.oracle.com/cve/ lassen sich alle verfügbaren Updates anhand der CVE-Nummer identifizieren, http://linux.oracle.com/errata/ listet alle (auch nicht sicherheitsrelevante) Updates auf.

Das folgende (gekürzte) Beispiel listet alle verfügbaren sicherheitsrelevanten Aktualisierungen für den Linux-Kernel auf:

```
[oracle@oraclelinux6 ~]$ yum list-security kernel*

Loaded plugins: refresh-packagekit, security
```

```
ELSA-2013-0567 Important/Sec. kernel-2.6.32-358.0.1.el6.x86_64

ELSA-2013-0830 Important/Sec. kernel-2.6.32-358.6.2.el6.x86_64

ELSA-2013-0911 Important/Sec. kernel-2.6.32-358.11.1.el6.x86_64

ELSA-2013-0567 Important/Sec. kernel-firmware-2.6.32-358.0.1.el6.noarch

ELSA-2013-0830 Important/Sec. kernel-firmware-2.6.32-358.6.2.el6.noarch

ELSA-2013-0911 Important/Sec. kernel-firmware-2.6.32-358.11.1.el6.noarch

ELSA-2013-2511 Important/Sec. kernel-uek-2.6.39-400.17.2.el6uek.x86_64

ELSA-2013-2524 Critical/Sec.  kernel-uek-2.6.39-400.24.1.el6uek.x86_64

ELSA-2013-2525 Important/Sec. kernel-uek-2.6.39-400.109.1.el6uek.x86_64

ELSA-2013-2511 Important/Sec. kernel-uek-devel-2.6.39-
400.17.2.el6uek.x86_64

ELSA-2013-2513 Important/Sec. kernel-uek-devel-2.6.39-
400.21.1.el6uek.x86_64

ELSA-2013-2525 Important/Sec. kernel-uek-devel-2.6.39-
400.109.1.el6uek.x86_64

ELSA-2013-2511 Important/Sec. kernel-uek-firmware-2.6.39-
400.17.2.el6uek.noarch

ELSA-2013-2524 Critical/Sec.  kernel-uek-firmware-2.6.39-
400.24.1.el6uek.noarch

ELSA-2013-2525 Important/Sec. kernel-uek-firmware-2.6.39-
400.109.1.el6uek.noarch

CVE-2012-2100  security       kernel-uek-headers-2.6.32-
300.39.2.el6uek.x86_64

ELSA-2013-2504 Moderate/Sec.  kernel-uek-headers-2.6.32-
300.39.4.el6uek.x86_64

CVE-2013-0190  security       kernel-uek-headers-2.6.32-
300.39.4.el6uek.x86_64

ELSA-2013-2512 Important/Sec. kernel-uek-headers-2.6.32-
300.39.5.el6uek.x86_64

ELBA-2013-2516 bugfix         kernel-uek-headers-2.6.32-
400.26.1.el6uek.x86_64

ELSA-2013-2520 Important/Sec. kernel-uek-headers-2.6.32-
400.26.2.el6uek.x86_64

ELSA-2013-2534 Moderate/Sec.  kernel-uek-headers-2.6.32-
400.29.1.el6uek.x86_64

updateinfo list done
```

Um weitere Details über einen bestimmten Update zu erhalten, kann das folgende yum-Kommando verwendet werden:

```
[oracle@oraclelinux6 ~]$ yum info-security CVE-2013-0190

Loaded plugins: refresh-packagekit, security


================================================================================

================================================================================

  Update ID : CVE-2013-0190

    Release :

       Type : security

     Status : final

     Issued : 2013-02-13

       CVEs : CVE-2013-0190

Description : The xen_failsafe_callback function in Xen for the Linux
kernel

            : 2.6.23 and other versions, when running a 32-bit

            : PVOPS guest, allows local users to cause a denial

            : of service (guest crash) by triggering an iret

            : fault, leading to use of an incorrect stack

            : pointer and stack corruption.

updateinfo info done
```

Um alle den Kernel betreffenden Sicherheits-Updates einzuspielen, kann das folgende Kommando verwendet werden:

```
[oracle@oraclelinux6 ~]$ sudo yum update --security kernel*

Loaded plugins: refresh-packagekit, security

ol6_UEK_base                                             | 1.2 kB     00:00

ol6_UEK_latest                                           | 1.2 kB     00:00

ol6_latest                                               | 1.4 kB     00:00

ol6_latest/primary                                       |  30 MB     00:23

ol6_latest
21705/21705

ol6_u4_base                                              | 1.4 kB     00:00
```

```
Setting up Update Process

Resolving Dependencies

Limiting packages to security relevant ones

ol6_UEK_latest/updateinfo                                  |  60 kB     00:00

ol6_latest/updateinfo                                      | 581 kB     00:00

6 package(s) needed (+0 related) for security, out of 6 available

--> Running transaction check

---> Package kernel.x86_64 0:2.6.32-358.11.1.el6 will be installed

---> Package kernel-firmware.noarch 0:2.6.32-358.el6 will be updated

---> Package kernel-firmware.noarch 0:2.6.32-358.11.1.el6 will be an update

---> Package kernel-uek.x86_64 0:2.6.39-400.109.1.el6uek will be installed

---> Package kernel-uek-devel.x86_64 0:2.6.39-400.109.1.el6uek will be
installed

---> Package kernel-uek-firmware.noarch 0:2.6.39-400.109.1.el6uek will be
installed

---> Package kernel-uek-headers.x86_64 0:2.6.32-300.39.1.el6uek will be
updated

---> Package kernel-uek-headers.x86_64 0:2.6.32-400.29.1.el6uek will be an
update

--> Finished Dependency Resolution

--> Running transaction check

---> Package kernel-uek-devel.x86_64 0:2.6.39-200.24.1.el6uek will be
erased

---> Package kernel-uek-firmware.noarch 0:2.6.39-200.24.1.el6uek will be
erased

--> Finished Dependency Resolution


Dependencies Resolved


================================================================================
=====

 Package              Arch    Version                     Repository
Size

================================================================================
```

```
=====

Installing:

 kernel               x86_64  2.6.32-358.11.1.el6         ol6_latest
26 M

 kernel-uek           x86_64  2.6.39-400.109.1.el6uek     ol6_UEK_latest
27 M

 kernel-uek-devel     x86_64  2.6.39-400.109.1.el6uek     ol6_UEK_latest
8.0 M

 kernel-uek-firmware  noarch  2.6.39-400.109.1.el6uek     ol6_UEK_latest
3.6 M

Updating:

 kernel-firmware      noarch  2.6.32-358.11.1.el6         ol6_latest
11 M

 kernel-uek-headers   x86_64  2.6.32-400.29.1.el6uek      ol6_latest
731 k

Removing:

 kernel-uek-devel     x86_64  2.6.39-200.24.1.el6uek      @ol6_UEK_base
27 M

 kernel-uek-firmware  noarch  2.6.39-200.24.1.el6uek      @ol6_UEK_base
4.4 M


Transaction Summary

================================================================================
=====

Install       4 Package(s)

Upgrade       2 Package(s)

Remove        2 Package(s)


Total download size: 77 M

Is this ok [y/N]: y

Downloading Packages:

(1/6): kernel-2.6.32-358.11.1.el6.x86_64.rpm             |  26 MB     00:20

(2/6): kernel-firmware-2.6.32-358.11.1.el6.noarch.rpm    |  11 MB     00:08

(3/6): kernel-uek-2.6.39-400.109.1.el6uek.x86_64.rpm     |  27 MB     00:21
```

```
(4/6): kernel-uek-devel-2.6.39-400.109.1.el6uek.x86_64.r | 8.0 MB     00:06
(5/6): kernel-uek-firmware-2.6.39-400.109.1.el6uek.noarc | 3.6 MB     00:02
(6/6): kernel-uek-headers-2.6.32-400.29.1.el6uek.x86_64. | 731 kB     00:00
--------------------------------------------------------------------------------
Total                                           1.3 MB/s |  77 MB     01:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Updating   : kernel-firmware-2.6.32-358.11.1.el6.noarch
1/10
  Installing : kernel-uek-firmware-2.6.39-400.109.1.el6uek.noarch
2/10
  Installing : kernel-uek-2.6.39-400.109.1.el6uek.x86_64
3/10
  Installing : kernel-2.6.32-358.11.1.el6.x86_64
4/10
  Updating   : kernel-uek-headers-2.6.32-400.29.1.el6uek.x86_64
5/10
  Installing : kernel-uek-devel-2.6.39-400.109.1.el6uek.x86_64
6/10
  Cleanup    : kernel-uek-firmware-2.6.39-200.24.1.el6uek.noarch
7/10
  Cleanup    : kernel-firmware-2.6.32-358.el6.noarch
8/10
  Cleanup    : kernel-uek-headers-2.6.32-300.39.1.el6uek.x86_64
9/10
  Cleanup    : kernel-uek-devel-2.6.39-200.24.1.el6uek.x86_64
10/10
  Verifying  : kernel-uek-devel-2.6.39-400.109.1.el6uek.x86_64
1/10
  Verifying  : kernel-uek-2.6.39-400.109.1.el6uek.x86_64
2/10
  Verifying  : kernel-uek-firmware-2.6.39-400.109.1.el6uek.noarch
3/10
  Verifying  : kernel-uek-headers-2.6.32-400.29.1.el6uek.x86_64
```

```
4/10

  Verifying  : kernel-firmware-2.6.32-358.11.1.el6.noarch
5/10

  Verifying  : kernel-2.6.32-358.11.1.el6.x86_64
6/10

  Verifying  : kernel-uek-devel-2.6.39-200.24.1.el6uek.x86_64
7/10

  Verifying  : kernel-firmware-2.6.32-358.el6.noarch
8/10

  Verifying  : kernel-uek-headers-2.6.32-300.39.1.el6uek.x86_64
9/10

  Verifying  : kernel-uek-firmware-2.6.39-200.24.1.el6uek.noarch
10/10


Removed:

  kernel-uek-devel.x86_64 0:2.6.39-200.24.1.el6uek

  kernel-uek-firmware.noarch 0:2.6.39-200.24.1.el6uek


Installed:

  kernel.x86_64 0:2.6.32-358.11.1.el6

  kernel-uek.x86_64 0:2.6.39-400.109.1.el6uek

  kernel-uek-devel.x86_64 0:2.6.39-400.109.1.el6uek

  kernel-uek-firmware.noarch 0:2.6.39-400.109.1.el6uek


Updated:

  kernel-firmware.noarch 0:2.6.32-358.11.1.el6

  kernel-uek-headers.x86_64 0:2.6.32-400.29.1.el6uek


Complete!
```

Nun wurden alle den Linux-Kernel betreffenden Sicherheits-Updates angewendet.

Allerdings werden diese erst wirksam, wenn das System mit dem neuen Kernel neu gestartet wurde! Ähnliches gilt auch für Systembibliotheken – erst nach Neustart der Applikation werden die neuen Bibliotheken tatsächlich verwendet.

Ressourcen

Weitere Informationen zum Thema RPM, yum und Updates für Oracle Linux:

- Kapitel über yum im Oracle Linux Administrator's Guide for Release 6: http://docs.oracle.com/cd/E37670_01/E41138/html/ol_yum.html
- Kapitel über yum im Oracle Linux Administrator's Solutions Guide for Release 6: http://docs.oracle.com/cd/E37670_01/E37355/html/ol_yum.html
- Oracle Linux Hands-on lab: Package management with RPM and yum: https://wikis.oracle.com/display/oraclelinux/Hands-on+Lab+-+Oracle+Linux+Package+Management
- OTN Blog-Artikel: How the Oracle Linux Update Channels are Structured https://blogs.oracle.com/OTNGarage/entry/how_the_oracle_linux_update

Kontaktadresse:
Lenz Grimmer
Oracle Deutschland B.V. & Co. KG
Riesstraße 25
D-80992 München

Telefon: +49 (0) 40 41267308
E-Mail lenz.grimmer@oracle.com
Internet: http://www.oracle.com/linux