USENIX Association

Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta

Atlanta, Georgia, USA
October 10–14, 2000

Knowing When To Say No

Allan Cantos
CTO, Acrylis, Inc.

You see a new piece of software posted on freshmeat, and you just know that it will solve that nagging problem you've been having with (insert name here). But what many of us don't realize, until it's far too late, is all the possible ramifications of installing (or deleting) even one simple piece of software. Allan Cantos, CTO at Acrylis (www.acrylis.com) will explore many real-world scenarios and pose questions that every Linux Administrator should ask prior to hitting the download button. Things like how much work is involved if I install that new software? What, if anything, will break in the process? Do I need to update any existing libraries on my system? What other key applications rely on those libraries? In the end, is it worth my time to do an upgrade?

In this seminar you will learn how to:

Understand Recursive Dependencies

While RPM gives some data on dependencies, what dependencies do the dependencies have? A detailed look in recursive dependencies of packages will be explored.

Manage Software Across All Systems

If a security alert on a particular piece of software arises, how can you quickly find it on all of your systems? Tools and techniques for remote system management and monitoring will be discussed.

Deal with Tarballs

Not all software is in RPM packages. In fact a vast majority of them are delivered, by the developer, as a compressed tar file. When installed on an RPM based system, it's not visible to the system's RPM database--where the resulting loss of visibility can create issues. Methods and techniques on dealing with tarballs will be discussed.

Plan for Installation

Once a decision is made to install new software, how do you scope out the "net list" of tasks necessary to do the upgrade on all effected systems?

About Allan Cantos

Allan Cantos, Chief Technology Officer for Acrylis Inc. has been in software development for 18 years, managing and developing systems software and software development tools. Before joining Acrylis, Allan managed the engineering organization of UniPrise Systems, Inc. Prior to that, Allan managed a prominent software development consulting operation. Earlier in his IT career, Allan was the manager of Integration Platforms at Apollo, leading the design and development of software frameworks for tool integration. Allan has focused on the design and development of large heterogeneous systems tools in the UNIX and NT markets. Allan has an MS in Computer Science from the University of California at Berkeley.

About Acrylis Inc.

Founded in 1998 and headquartered in North Chelmsford Massachusetts, Acrylis Inc., is an emerging leader in the open-source, software management industry. Acrylis provides system administrators with Internet-delivered tools and services for faster, more reliable software management. The company has developed WhatifLinux, an Internet based subscription service that proactively monitors and manages the software assets running on networks of dynamic, Linux servers. Acrylis takes a unique approach to delivering critical Linux software information. By focusing their efforts on deploying autonomous agents that work in conjunction with their Knowledge Base, Acrylis is creating a community of open-source software know-how, which is designed for quicker, more reliable open- source software management. For more information, visit www.WhatifLinux.com.

For Additional Information Please Contact:

Chris McCoin
McCoin & Smith Communications LLC
508-881-0095
chris@mccoinsmith.com

Knowing When To Say No

Allan Cantos
CTO, Acrylis, Inc.

Introduction

Linux acceptance in corporate enterprises has moved from being on one or two evaluation systems to being used on many systems. While the benefits of Linux have been well documented, the software management issues are another matter. Specifically, how does an administrator manage and maintain software on multiple Linux systems given the dynamic change in free/open source software development.

Research done by Acrylis, Inc. shows that administrators spend between 30 minutes to an hour a day keeping up with the changes in free/open source software. They spend an additional 30 minutes to one hour evaluating how this software applies to their systems, and accessing what needs to be done to implement what they've just researched.

This paper outlines some of the issues and possible solutions.

Understanding Recursive Dependencies

Not long ago, new software was added to UNIX and Linux systems using 'tarballs' or more specifically, tape archive record files. As we know, tar files simply read and write what a directory of files has, and places them in the relative place on the system. Putting software in the correct directory is a good start, but much additional time is needed to get software configured to work properly. This includes setting environment variables, editing scripts and resource files, creating symbolic links, and more. The whole process is further complicated when dependencies arise when running the software for the first time. The administrator discovers a shared library was missing, or needs to be upgraded.

To resolve the issue of properly configured software, Marc Ewing and Eric Troan developed the Red Hat Package Manager (RPM). RPM addresses some of the complexities in installing, upgrading, and removing software. Today, RPM has become a standard, used by many of the Linux distributions in addition to developers wanting to deliver software to users.

With very few exceptions, all software depends on installed components resident on a target system. These dependencies are outlined in the dependency tag of the RPM Spec file that the target system uses to compare to the system's RPM database. For very simple packages, this system works well, but when a dependency has its own dependencies, or when one or two of the installed components, it doesn't quite work well. Other problems arise when these dependencies conflict with each other, or need to be updated to fulfill their support role.

Problems with Recursive Dependencies and RPM

The main problem with RPM installation today is that it doesn't the installer whether the software package being installed requires software that has subsequent dependencies. Administrators running into this problem can get caught up in a *version tango*, manually going down each dependency to find out what needs to be installed, what needs to be updated, or what is in conflict and why. The result is time consumed in installing software that could be spent doing other work.

Common remedies employed by Administrators faced with recursive dependencies includes using a --nodeps install, or using tarballs to install software. While these are the quickest paths to getting software installed, it has a high risk of not working correctly. What's left is software invisible to the other software in the RPM database, giving rise to other software incompatibilities and other hidden *gotchas* later.

One of problems causing recursive dependency is that information isn't filled out properly or completely in the package description tags, yielding an incorrect analysis from the RPM query. This problem makes it difficult to do any package equivalency analysis. Furthermore if an administrator installs a SuSE package on a Red Hat system the problem might get worse. More detail is provided in the table below, which contains that dependency information for the same version of Apache web server, but delivered by two different vendors:

Red Hat: apache-1.3.9-8-i386 Dependencies	TurboLinux: apache-1.3.9-6-i386 Dependencies
/etc/mime.types	/etc/mime.types
/sbin/chkconfig	/sbin/chkconfig
/bin/sh	/bin/sh
/usr/bin/perl	/usr/bin/perl
/bin/mktemp	
/bin/rm	
mailcap	
grep	
textutils	
ld-linux.so.2	ld-linux.so.2
libc.so.6	libc.so.6
libcrypt.so.1	libcrypt.so.1
libdb.so.3	libdb.so.3
libdl.so.2	libdl.so.2
libm.so.6	libm.so.6

Source: www.whatiflinux.com

These are essentially the same package, but Red Hat's release has five dependencies not found in the TurboLinux release. Which one is correct? Can the TurboLinux package be installed on the Red Hat system, and visa versa?

Some solutions to this problem exist. Debian already solves this by including recursive dependency information in its package descriptions, which is something RPM may implement in the future. The Linux Standards Base is an effort to solve many of the interoperability problems with Linux, including installing RPMs across distributions.

Managing Software Across All Systems

The complexity of software management on Linux systems increases when it is extended to dozens of systems stretched out over a corporate enterprise. If a security alert on a particular piece of software arises, how can you quickly find it on all of your systems? And once found, how easy is it to update?

Complicating the problem is the necessity for other people in the organization to have some choice in their system setup, and require access (root or sudo) to the system. This creates the possibility of system differences that need to be reconciled and managed when necessary updates need to be done.

One unfortunate solution to the problem of managing multiple systems is to mandate system configuration, essentially making all machines the same. This makes it easy to manage one system, and then push the results to all other systems. The disadvantage of this method is that systems can no longer be specialized for tasks (the Web server and the Samba server for example). Common Practice involves reducing the software configuration to only necessary daemons for increased security and reliability of the system. It also doesn't provide the necessary flexibility relevant to today's rapidly changing environment.

The choice for administrators for managing software across many systems includes scripting and record keeping (either manual or with a database). RPM lends itself to scripting because of its command line interface. Graphical applications such as Gnorpm can also be used when run remotely via X. The ultimate solution would allow an administrator to view all software on a specific system, and to view specific software over all systems.

Dealing with Tarballs

As mentioned earlier in this paper,not all software is in RPM packages. In fact, a vast majority of software is delivered by the developer as a compressed tar file. When installed on an RPM based system, the software is invisible to the system's RPM database. This loss of visibility can create issues with future software installations. This is especially true when dealing with dozens of systems, not all of which are under the control of the administrator.

Aside from auditing systems on a file-by-file basis, the only option for administrators is to get the system under control as soon as possible. Administrators can rebuild tars into RPMs using rpmbuilder. This application allows administrators to get packages under control before installation.

Plan for Installation

Once a decision is made to install new software, it's important to understand how to scope out the "net list" of tasks necessary to do the upgrade on all effected systems. The process is presented below:

1. **Find the most up to date, or most stable package.**

The many sources for this information present a challenge to administrators. Various source include the mirror sites, rpmfind.net. Red Hat charges for a service that gives customers priority access to FTP servers.

2. **Research the package.**

Review if the software has any outstanding security alerts. Check Bugtraq, Security Focus, X-Force, to see any issues.

3. **Verify package against public keys for vendor.**

Verifying a package can't occur until the software is downloaded. Can your source of the binary be trusted?

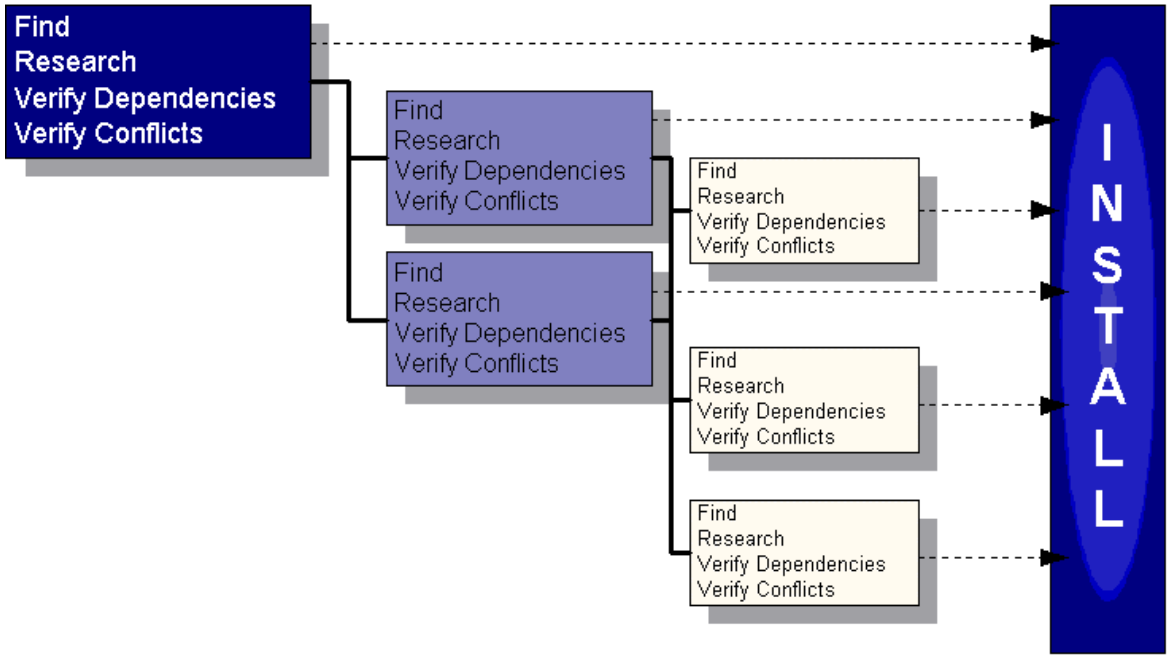
4. **Are there dependencies?**

If so, repeat above process until completed.

5. **Will installing a dependency impact other software running on the system?**

A risk assessment needs to be done whenever new libraries or other components are installed. Adding a new feature could possibly clobber an existing one.

If a problem is found (security or otherwise) during the installation process, the process needs to be repeated with the next available package. This is a very time consuming process, which lends itself to automation. Here's a graphical example of the process:



Conclusion

While current software installation and management systems have served us well up to this point, extending them to manage machines across the enterprise, with the level of dynamic change in the underlying software brings a whole new set of challenges. Undoubtedly, these challenges will be met by new methods in the near future.

Sources:

Maximum RPM, Edward C. Bailey, SAMS Publishing copyright 1997by Red Hat Software Inc.
RPM Builder,
http://sourceforge.net/projects/rpmbuilder/
Linux Standards Base: http://linuxbase.org