Lab 2.3 : Introduction to Packages

Objectives

- Introduction to Packages
- Types of Package Managers
- Community-Run Repositories
- Flatpaks and Snap

Topic 1: Introduction to Packages

Installing software on the Linux platform utilizes a 3-layer process for the distribution, installation, and maintenance of any particular software. These three layers are Packages, Repositories, and Package Managers. Let's talk about these components in more detail so that we have a better understanding of each one's purpose:

Packages:

Packages are containers that house the application itself, as well as any dependency requirements, configuration files, or other scripts. Using packages instead of the raw application promotes better system management, preventing users from running software with missing dependencies, or packages that are out-of-date and missing the latest security features.

Repositories:

Repositories, otherwise known as repos, are the warehouse in which packages are stored. Developers can upload their packages to the repository, where the packages are vetted and tested for malware, and are required to meet certain requirements. While Linux distributions often have their own repositories that are managed by the operating system developers, there are some examples of user-created repositories like Git and the AUR.

Package Managers:

Package Managers are software that is typically pre-installed onto the operating system that makes the management of packages more user-friendly. Checking for updates, notifications regarding dependency incompatibility, as well as the ability to upgrade and install new packages are presented in a format that is much more intuitive for the average user. It's worth noting that different package managers are compatible with different types of packages, so what packages can be installed will be limited to the package manager you have available on your distribution.

Topic 2: Types of Package Managers

Depending on the Operating System that is installed, there are a variety of different package managers that are available. Below is a list of some of the most common package managers.

APT

The Advanced Package Tool (APT) package manager is perhaps the most commonly utilized package manager, as it comes from Debian, the grandfather system in Linux. Common systems that run this package manager include Debian, Ubuntu, and Kali Linux. The APT package manager provides a user interface for the dpkg manager, which allows packages with a .deb file format to be installed. Sometimes the command line requires ‘apt-get’, but newer versions support ‘apt’.

Ex. ‘sudo apt install git’

DNF

Dandified Yum (DNF) is the package manager associated with Fedora, CentOS, and other RHEL-based operating systems. It provides the user interface for the RPM Package manager, which allows the installation of packages with .rpm format to be installed. In older versions, this package manager was referred to as Yellowdog Updater, Modified (YUM) and used the ‘yum’ command to install packages. The yum commands are still supported by modern versions of DNF.

Ex. ‘sudo dnf install git’, ‘sudo yum install git’

PACMAN

Pacman, a shorthand for ‘Package Manager’, is the Arch-based package manager utilized by systems such as Arch linux and Manjaro. It operates on packages that are stored using .tar.xz file formats, and is unique in that the packages are stored in a compressed format that reduces overall space and combines the building and managing process.

Ex. ‘sudo pacman install git’

SLACKPKG

Slackpkg is the default package manager for Slackware distributions. It allows the user to install any package with a .tgz or .txz file format. Slackware is unique in that it provides commands such as installpkg, removepkg, and upgradepkg for more advanced features, rather than bundling these tasks into a single command using arguments.

Ex. ‘slackpkg install git’

Topic 3: Community-Run Repositories

Community run repositories are publicly available repositories that contain packages that are managed by a group of users outside of the developers of the system. Perhaps the most common community-run repository is the AUR, or Arch User Repository. User driven repositories allow for further customization, as users can create their own versions of software and upload them onto the repo. Users install these packages using an alternative package manager by following the guide provided by the community-run repository managers. In the case of the AUR, users can vote on packages that, if popular enough, will be accessible via pacman.

However, many of the packages uploaded onto user repositories are not vetted or tested extensively. As a result, installing packages from locations such as the AUR can introduce new security vulnerabilities, cause dependency issues, or outright package conflicts that can seriously damage the system. In some cases, packages are not maintained, and the use of a separate package manager will not detect dependency problems. Later on, a regular update of the system packages may introduce a new vulnerability or dependency issue that did not previously exist.

Community-run repositories are a fantastic resource, but in general it’s best to avoid using them extensively, as it will save you trouble in the long run. Although it can be more work, utilizing the built-in package manager and manually making your own configuration changes will ensure better stability long-term.

Topic 4: Flatpaks and Snap

Other common package distribution management tools are Flatpaks, and the Snap Store. These are platforms that operate outside of the operating system's base package manager, but are also not quite community-run repositories.

- Flatpak

  Flatpak is a package manager that works across several distributions and installs flatpak apps that are hosted on the Flathub repository. This repository is managed by the owners of Flatpak, and is unique in that it provides a platform that sandboxes all running applications in an effort to increase security. Flatpaks also take up less space than most other packages once the framework is installed, which encourages the use of many Flatpaks.

- Snap

  Snap is another package manager that works across several linux distributions through the use of packages called 'snaps'. You can install snaps independently, or use the Snap Store, a GUI service created and managed by the developer of Snap. For security, snaps utilize AppArmor, a security measure built into several major linux distributions.

For both of these alternative package managers and others, it's important to remember that untrusted applications should never be run, sandboxed or not, and that there have been several documented sandbox escape methods found in the past. In the case of Snaps, not having AppArmor active will result in any snap having the same level of system access as regular packages from the system package manager, which introduces several vulnerability concerns.

Similarly to the community-run repositories, these are fantastic resources if utilized sparingly and managed well. If left unchecked, they may cause problems in the future.

Topic 5: Review

---

1. What command should be used to install Python on Ubuntu?
2. Why are packages used over downloading raw applications?
3. Can you use pacman on Fedora? Why or why not?
4. Are Snaps secure? Explain your answer.
5. What are the different layers used in the Linux platform?
6. What file types can be installed using SLACKPKG?
7. When should you use a community based package manager over a fault?
8. Why would you use Flatpack?
9. Which layers would you be accessing by downloading with Git?
10. When would you use ‘apt-get’ compared to ‘apt’?

Resources

- https://aur.archlinux.org/
- https://www.flatpak.org/
- https://snapcraft.io/store