LESSON NOTES

Intro to Linux

System Management

1.6.1 Software Installation Methods

Lesson Overview:

Students will:

- Understand how software is installed and updated in a Linux system

Guiding Question: How is software installed and kept up to date?

Suggested Grade Levels: 9 - 12

Technology Needed: None

CompTIA Linux+ XK0-005 Objective:

1.6 - Given a scenario, build and install software

- Package management
  - DNF
  - YUM
  - APT
  - RPM
  - dpkg
  - ZYpp
- Sandboxed applications
  - snapd
  - Flatpak
  - AppImage
- System updates
  - Kernel updates
  - Package updates

This content is based upon work supported by the US Department of Homeland Security's Cybersecurity & Infrastructure Security Agency under the Cybersecurity Education Training and Assistance Program (CETAP).

Software Installation Methods

Packet Management

Package management is a critical aspect of managing software on Linux-based systems. Different Linux distributions use various package management tools to handle software installation, updates, and removal. CompTIA references DNF, YUM, APT, RPM, dpkg, and ZYpp.

DNF (Dandified YUM) is the next-generation package manager used in Red Hat-based Linux distributions such as Fedora and CentOS. This replaces the older *YUM* (Yellowdog Updater Modified) package manager. Both DNF and YUM are used to install, update, and remove software packages. They resolve dependencies and can automatically download and install required packages.

APT (Advanced Package Tool) is the package manager used in Debian-based Linux distributions such as Debian, Ubuntu, and Kali. It manages packages and dependencies efficiently using commands like **`apt-get`** or **`apt`** to install, update, and remove software packages while also handling package dependency resolution.

RPM (Red Hat Package Manager) is a low-level package management tool used primarily in Red Hat-based distributions, including Red Hat Enterprise Linux (RHEL) and CentOS. RPM is used for installing, querying, and managing individual RPM packages. It does not handle dependency resolution though, so users often rely on YUM or DNF to manage packages.

The package manager used at the core of Debian-based Linux distributions is ***dpkg***. This is responsible for installing, configuring, and removing software packages. Users typically interact with dpkg through higher-level package management tools like APT. dpkg manages individual .deb package files and ensures proper installation.

ZYpp (libzypp) is the package manager used in openSUSE and SUSE Linux Enterprise. It is designed to handle package management, dependency resolution, and system updates. ZYpp provides command-line tools for managing packages and repositories using commands like **`zypper`** to install, update, and remove software packages.

These package management tools are essential for maintaining software on Linux systems. The tool(s) depends on the Linux distribution being used. Users should become familiar with the package management system relevant to their chosen Linux distribution to efficiently manage software packages and keep their systems up to date.

Sandboxed Applications

Sandboxed applications are a way to package and distribute software with isolated environments to enhance security, compatibility, and ease of use. CompTIA references three sandboxed application formats: snapd, Flatpak, and AppImage.

A package manager developed by Canonical for Ubuntu and other Linux distributions is called ***snapd*** (Snap Package Manager). This allows developers to package applications along with their dependencies in a single snap package, providing sandboxing and automatic updates. Some key features of snapd are isolation, automatic updates, and cross-distribution. Snap packages run in isolated environments called "snaps," which have restricted access to the host system. Snaps can be updated automatically and are designed to work across various Linux distributions, providing software compatibility.

Flatpak (formerly xdg-app) is a sandboxing framework and package manager that allows developers to create and distribute applications with their own runtime and libraries, ensuring consistent behavior across different Linux distributions. Flatpak applications run in isolated environments called "runtimes," which contain all necessary dependencies. Applications bundle their runtime, reducing dependency on specific system libraries and ensuring compatibility. Flatpak uses a centralized repository for distribution, making it easy to discover and install applications.

AppImage is a format for packaging applications as a single, self-contained executable file. It doesn't require installation or system modifications, making it a portable and versatile solution. AppImages are single binary files that can run on various Linux distributions. They include all necessary libraries, reducing compatibility issues. Users have direct control over AppImages, and they can easily move, run, or delete them as needed.

Sandboxed applications aim to simplify software distribution and enhance security by isolating applications and their dependencies. Each format has its own unique features and strengths, making them suitable for different use cases and preferences. Users and developers can choose the format that best fits their needs and the distribution they are using.

System Updates

System updates are essential for maintaining the stability, security, and performance of a computer system. They typically include two main types of updates: kernel updates and package updates.

Kernel updates involve updating the core of the OS, known as the kernel. The kernel is responsible for managing hardware resources and providing essential system services. Kernel updates often include patches for security vulnerabilities, protecting the system from potential threats. Kernel updates may improve system performance by optimizing resource management and fixing bugs. Newer kernels often include support for newer hardware devices. Kernel updates are usually provided by the OS's package manager and can be installed through system update commands like **`apt upgrade`**.

Package updates involve updating software packages and applications installed on the system. These packages may include system utilities, libraries, and desktop applications. Package updates often include security patches for vulnerabilities in software components. Updates may introduce new features, enhancements, or bug fixes to improve the functionality of applications. Keeping packages up to date also ensures compatibility with other software and system components. Package updates are managed by the system's package manager and can be updated through system update commands like **`apt update && apt upgrade`** to fetch and install the latest package updates.

System updates are crucial for maintaining a secure, stable, and up to date computer system. Kernel updates focus on the core of the OS, while package updates cover a wide range of software components. Regularly applying both types of updates is essential for keeping the system secure, improving performance, and ensuring software compatibility.