Lahore Garrison University

Shell Scripting

Submitted by:

Armoghan-ul-Mohmin

Roll no 010

Submitted to:

Sir Muhammad Hamza

Catalog

Security Patch Manager

PROJECT OBJECTIVES

The core objective of the Security Patch Manager project is to create a user-friendly and efficient Bash script. This script aims to automate the often time-consuming process of managing system security updates. By providing administrators with a streamlined tool, the script enhances the overall security posture of Linux-based systems.

SCRIPT OVERVIEW

Feature

1. User-Friendly Interface:

The script incorporates an intuitive and aesthetically pleasing interface. It includes a dynamic ASCII banner, colour-coded messages, and interactive menu options, making it accessible to users with varying levels of technical expertise.

2. Root Privilege Handling:

To ensure the script can perform necessary system operations, it checks for root privileges. Users have the option to run the script with elevated permissions using the **--root** option.

3. Update Listing:

Users can easily view available security updates in a neatly formatted table. The use of colour-coded text enhances visibility and simplifies the identification of relevant information.

4. Live Progress Bar:

When applying security updates, the script provides real-time feedback through a live progress bar. This feature keeps users informed about the ongoing update process.

5. Autoclean and Autoremove:

The script includes options to execute **apt autoclean** and **apt autoremove** with improved formatting. This enhances the overall user experience by presenting results in a more organised and readable manner.

Functions

- **print_message(COLOUR, MESSAGE)**: Displays coloured messages on the terminal.
- **exit_message()**: Displays exit message and exits the script.
- **show_version():** Displays the script version.
- **press_enter():** Waits for Enter key press to continue.
- **print_banner():** Prints the contents of an ASCII banner.
- **print_linux_util_banner():** Prints the Linux-Util banner with project information.
- **RUN_AS_ROOT():** Checks if the script is run with root privileges.
- **internet_connection():** Checks for a working internet connection.
- **list_security_updates():** Lists available security updates.
- **apply_security_updates():** Applies security updates with a progress bar.
- **perform_autoclean():** Performs 'apt autoclean'.
- **perform_autoremove():** Performs 'apt autoremove' with improved formatting

Additional Functions

· **Trap Ctrl+C**: The script incorporates a mechanism to trap the **Ctrl+C** signal. This ensures a graceful exit in response to user interruptions, providing a positive user experience.

· **Version Display**: Users can check the script version using the **-v** or **–version** option. This feature enables users to confirm the script's current version.

· **Help Menu**: The **–h** or **--help** option provides users with a comprehensive help menu. This menu outlines available options and their functionalities, ensuring users can make informed decisions.

DEPENDENCIES

The following dependencies are required to run the Security Patch Manager script:

1. **awk**: A programming language and utility for pattern scanning and processing. It's used in various parts of the script for text processing.
2. **column**: A command-line utility to format its input into multiple columns. It's used to format the output of certain commands for better readability.
3. **ping**: A command-line utility to test the reachability of a host on an IP network. In the script, it's used to check for an active internet connection.
4. **sudo**: A command-line utility that allows a permitted user to execute a command as the superuser or another user. It's used to run certain commands with elevated privileges.
5. **apt**: The package management tool for Debian-based Linux distributions like Ubuntu. It's used to update, upgrade, and manage software packages.

Make sure these dependencies are installed on your system. Most Linux distributions come with these utilities pre-installed, but if any of them are missing, you can install them using your package manager. For example, on Ubuntu, you can install missing dependencies.

Configuration Options

- **Root Access**: Use **--root** or **-r** option to run the script as root.
- **Version Display**: Use **--version** or **-v** option to display script version.
- **List Available Updates**: Use **--list** or **-l** option to display available security updates.
- **Apply Updates**: Use **--apply** or **-a** option to apply security updates.
- **Auto Clean**: Use **--clean** or **-c** option to perform 'apt autoclean'.
- **Auto Remove**: Use **--remove** or **-d** option to perform 'apt autoremove'.
- **Help**: Use **--help** or **-h** option to display the help menu.

Testing Procedures

- **Unit Testing**: Verify individual functions and components.
- **Integration Testing**: Ensure seamless module interaction.
- **Regression Testing**: Confirm previous features still work.
- **User Acceptance Testing (UAT)**: Validate usability with stakeholders.
- **Performance Testing**: Assess system performance under load.
- **Security Testing:** Test for vulnerabilities and secure coding.
- **Compatibility Testing:** Ensure compatibility across platforms.

- ❖ **Documentation Testing**: Review documentation for accuracy.

Documentation Standards

README.md:

- Provide a brief overview, installation instructions, usage guidelines, and contribution guidelines.

Code Comments:

- Use comments to explain complex logic or provide context.

Inline Documentation:

- Document functions, classes, and modules with clear information about parameters, return values, and exceptions.

External Documentation:

- Provide user guides if necessary, and host them on a reliable platform.

Versioning:

- Document changes and updates between versions, following a versioning scheme like Semantic Versioning (SemVer).

Consistency:

- Maintain consistent writing style, formatting, and terminology.

Accessibility:

- Ensure documentation is accessible to all users.

Version Control

The source code for this project is managed using Git, a distributed version control system. The repository is hosted on GitHub.
You can find the latest source code, report issues, and contribute to the project on Security-Patch-Manager.
You can view the version history, including release notes and changes, on the GitHub Releases page. For more insights, tutorials, and updates about this project, visit Armoghan-Blogs.

Contributing Guidelines

Thank you for considering contributing to this project! Your participation helps make it better for everyone. For detailed instructions on how to contribute, please refer to the Contributing Guidelines section in our GitHub repository.

Licence information.

This project is licensed under the MIT License.