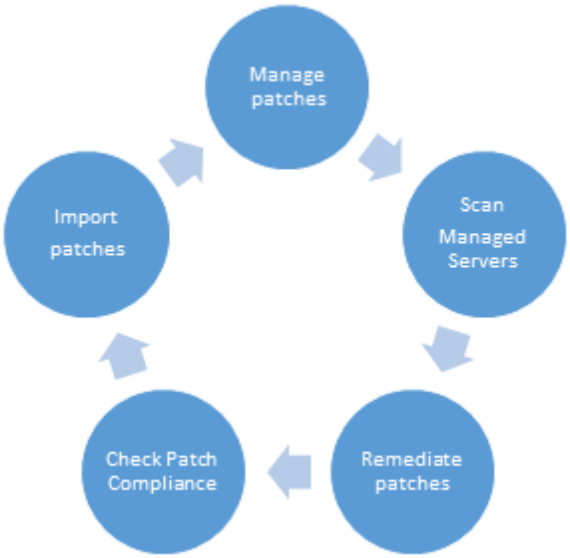
PATCH MANAGEMENT IN LINUX

1) What is Linux patching?

Linux patching (or patching in Linux) is the process of applying patches (or software codes) to fix vulnerabilities or to add new features to the Linux endpoints across your network. Patches are updates that contain changes in source code and are installed into the existing software program.

Here's what patches can do:

- **Address new security vulnerabilities.**
- **Integrate new features.**
- **Address a specific bug or flaw.**
- **Improve Operating system/software stability.**
- **Install new drivers.**



2) Why is patch management in Linux important?

Patching your Linux systems is crucial to prevent threat actors from exploiting vulnerabilities in them. It strengthens data security and fends off recurring attacks. Some patches also add new features and functions to the applications**.** Unpatched systems are one of the easiest entry points for hackers to gain access to the network.

So, patches to an Operating system are essential to keep the systems up to date, stable and safe from malware and other security threats.

3) What is Linux update management?

The process of Linux update management involves managing and applying patches to the Linux-based operating systems and to the applications running on Linux computers. Thus, ensuring the Linux environment is secure and up-to-date.

4) How do I use patch management in Linux?

Linux Patching is crucial as it helps keep your Linux environment secure and up-to-date. Patch Manager Plus is a reliable Linux patch management tool that helps you achieve seamless Linux patching in your network. You can implement an efficient approach to managing patches in your environment by choosing between manual and automatic patching methods.

How to Patch Your Linux Systems Manually?

Even when patch automation and live patching is in place, manual updates are occasionally necessary. For example, after a failed update, administrators may need to manually patch the system. Manual updates might also be necessary in a testing environment. The commands to update Linux depend on your distribution, but here are the commands for some common distributions.

For Red Hat Linux distributions (e.g. RedHat, CentOS, Oracle), the following commands check for updates and patch the system:

```
yum check-update

yum update
```

Patch management best practices

Unpatched and out-of-date systems can be a source of compliance issues and security vulnerabilities. In fact, most vulnerabilities exploited are ones already known by security and IT teams when a breach occurs. Patching strategy should also account for cloud and containerized resources, which are deployed from base images. Ensure that base images are compliant with organization-wide security baselines. As with physical and virtualized systems, scan and patch base images regularly

Automating patch management

Implementing a vigilant patch management policy takes planning, but patch management solutions can be paired with automation software to improve configuration and patch accuracy, reduce human error, and limit downtime.

Automation can drastically reduce the time IT teams spend on repetitive tasks, like identifying security risks, testing systems, and deploying patches across thousands of endpoints. Managing

these time-consuming processes with reduced manual input frees up resources and enables teams to prioritize more proactive projects.

How Does Automated Linux Patch Management Software Work?

Automated patch management operates on several layers, and vulnerability scanning is one. To avoid becoming the next newsworthy data breach, organizations must do vulnerability scans on every device.

Vulnerability scans identify if patches are missing, so administrators can deploy them as soon as possible. There are a few good vulnerability scanners available that make this first step much more efficient and convenient. These scanners are:

- Qualys
- Nessus
- Rapid7

With a scan complete, it's time for patch management tools to take over. Several tools on the market make patching much more convenient for administrators.

Better tools report on successful and failed patches so that administrators know when manual updates are also necessary. Patch management tools, therefore, deliver a comprehensive update on the current cybersecurity health of the enterprise environment. A few tools available to manage patches include:

- **Yum – used in Red Hat Enterprise Linux (RHEL)**
- Ansible
- Puppet
- SaltStack
- Chef
- Spacewalk

The above tools' primary advantage is improved organization, because they download updates and then report the results to administrators. In that process, the right tool can minimize the disorganization that can occur when patch management is handled across a large environment.

Administrators can also schedule patches, choose their own deployment policies, test, and then approve updates before deployment.

Use the check-update command to non-interactively check for outstanding updates on your server:

```
[root@rhel77 ~]# yum check-update
```

RHEL 7.9 was the most recent version of RHEL 7 and doing a yum update would take this RHEL 7.7 server to RHEL 7.9:

```
[root@rhel77 ~]# yum -y update
```

After a successful reboot, check the RHEL version to confirm that the server is updated to 7.9:

```
[root@rhel77 ~]# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.9 (Maipo)
```

Reverting the updates

First, check the history of the transaction with the yum command:

```
[root@rhel77 ~]# yum history
Loaded plugins: product-id, search-disabled-repos, subscription-manager
ID     | Login user             | Date and time    | Action(s)      | Altered
-------------------------------------------------------------------------------
     8 | root <root>            | 2020-11-01 23:10 | I, O, U        |  157 EE
```

The Action(s) and Altered columns give information about what changes occurred with this transaction.

Action(s):

- I - New package Installed
- O - Package is Obsoleted
- U - Package is Updated

Altered:

- 157 packages were altered
- EE - There were some errors/warnings in the transaction

Using yum history packages-list shows the changes that happened from that package's point of view.

As systemd is the first process started in RHEL versions 7 and above, that package is protected with /etc/yum/protected.d/systemd.conf:

```
[root@rhel77 ~]# cat /etc/yum/protected.d/systemd.conf
systemd
```

Revert the last transaction with yum history undo:

[root@rhel77 ~]# yum history undo last

Check the version of RHEL and the kernel:

[root@rhel77 ~]# cat /etc/redhat-release

Red Hat Enterprise Linux Server release 7.7 (Maipo)

[root@rhel77 ~]# rpm -q kernel

kernel-3.10.0-1062.el7.x86_64

kernel-3.10.0-1160.2.2.el7.x86_64

[root@rhel77 ~]# uname -r

3.10.0-1160.2.2.el7.x86_64

- identify the transaction ID that we want to 'undo'

```
# yum history
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
ID     | Login user            | Date and time    | Action(s)      | Altered
-------------------------------------------------------------------------------
     8 | root <root>           | 2011-10-03 14:40 | Install        |    1
     7 | root <root>           | 2011-09-21 04:24 | Install        |    1 ##
     6 | root <root>           | 2011-09-21 04:23 | Install        |    1 ##
     5 | root <root>           | 2011-09-16 13:35 | Install        |    1
     4 | root <root>           | 2011-09-16 13:33 | Erase          |    1
     3 | root <root>           | 2011-09-14 14:36 | Install        |    1
     2 | root <root>           | 2011-09-12 15:48 | I, U           |   80
     1 | System <unset>        | 2011-09-12 14:57 | Install        | 1025
```

- The transaction ID we are interested in is '8', so move forward with undo step. If you want to see additional information to verify this is transaction you are interested in, use yum history info 8 prior to performing the undo

```
# yum history undo 8
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
Undoing transaction 8, from Mon Oct  3 14:40:01 2011
    Install screen-4.0.3-16.el6.i686
Resolving Dependencies
--> Running transaction check
---> Package screen.i686 0:4.0.3-16.el6 will be erased
--> Finished Dependency Resolution


Dependencies Resolved
================================================================================
 Package        Arch      Version          Repository          Size
================================================================================
Removing:
 screen         i686      4.0.3-16.el6     @rhel-6-server-rpms     783 k

<snip>

Removed:
  screen.i686 0:4.0.3-16.el6
Complete!
```

A minor version is a release of **RHEL** that does not (in most cases) add new features or content. It focuses on solving minor problems, typically bugs or security issues. Most of what makes a

specific minor version is included in the kernel, so you will need to find out which kernels are supported as part of the minor version you are targeting. Let's check if the required kernel packages "**kernel-3.10.0-862**" is installed or not, using the following yum command.

```
# yum list kernel-3.10.0-862*
```

```
[root@server2 ~]# yum list kernel-3.10.0-862*
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.ucu.ac.ug
 * elrepo: ftp.nluug.nl
 * epel: epel.mirror.liquidtelecom.com
 * extras: mirror.ucu.ac.ug
 * updates: mirror.ufs.ac.za
Installed Packages
kernel.x86_64                    3.10.0-862.el7                    @anaconda
[root@server2 ~]#
```

```
yum install kernel-3.10.0-862.el7
```

Once the kernel installation is compete, to apply the changes, you need to reboot the system.

Then downgrade the **redhat-release** package to complete the process. The command below targets the latest minor version that is lower than the current running one, such as from **7.6 to 7.5**, or from **7.5 o 7.4**.

```
# yum downgrade redhat-release
```

Finally, confirm the downgrade by checking the contents of **/etc/redhat-release** using the cat command.

```
# cat /etc/redhat-release
```

```
[root@server2 ~]# cat  /etc/redhat-release
CentOS Linux release 7.5.1804 (Core)
[root@server2 ~]#
```