Linux (RHEL/CentOS) Patching Process

What are patches?
Patches are new or updated lines of code that determine how an operating system, platform, or application behaves. Patches are usually released as-needed to fix mistakes in code, improve the performance of existing features, or add new features to software. Patches are not newly compiled OSs, platforms, or applications patches are always released as updates to existing software.

Patches can also impact hardware like when we released patches that altered memory management, created load fences, and trained branch predictor hardware in response to the Meltdown and Spectre attacks of 2018 that targeted microchips.

Because modifications like these are usually quicker to distribute than minor or major software releases, patches are regularly used as network security tools against cyber-attacks, security breaches, and malware vulnerabilities that are caused by emerging threats, outdated or missing patches, and system misconfigurations.

Why to manage patches?
Because patching without a clearly defined patch management process can get messy.
Enterprise IT environments can contain hundreds of systems operated by large teams requiring thousands of security patches, bug fixes, and configuration changes. Even with a scanning tool, manually sifting through data files to identify systems, updates, and patches can be onerous.

Patch management tools help generate clear reports on which systems are patched, which need patching, and which are noncompliant.

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Unpatched and out-of-date systems can be a source of compliance issues and security vulnerabilities.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.

1. Identify systems that are noncompliant, vulnerable, or unpatched. Scan systems daily.
2. Prioritize patches based on the potential impact. Calculate risk, performance, and time considerations.
3. Patch often. Patches are usually shipped once a month or sooner.
4. Test patches before placing them into production.

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. When patching a base image, rebuild and redeploy all containers and cloud resources based on that image.

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.
For example, a handful of Red Hat Ansible Automation Platform modules can automate portions of patching processes, including invoking HTTP patch methods, applying patches using the GNU patch tool, and applying (or reverting) all available system patches.

Linux (RHEL/CentOS) Patching Process

For many organizations, multiple servers work together for one customer, and these servers since their functions are intertwined must be rebooted in a specific order when patches are deployed. With Ansible Automation Platform, the Ansible Playbook ensures this happens correctly and consistently, so IT teams don't have to.

Linux Server Patching Procedure:
Standard patching procedure can vary organisation to organisation. Some of the common steps involved in the standard patching practices in many organisations are as follows:

1. Check the patches availability. # dnf repolist all ; dnf check-update

2. Get the schedule down-time from the concern teams and clients.

3. Raise the RFC or change request as per the schedule in the change management tool (Service Now, Assist etc.) involving the clients, higher management and all the other operations teams like DB, Backup, Middleware, VMWare, UNIX etc.

4. Collect the server pre-patch (pre-checks) details and document them.

You can create a script for the same like below:
# vim precheck.sh

```
#!/bin/bash

echo -e "\nDate & Time Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
date >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nMounted Filesystems:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
df -Th >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nBlock IDs Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
blkid >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nDisks Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
fdisk -l >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nBlock Storage Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
lsblk >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nVolume Groups Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
vgdisplay >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nLogical Volume Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
lvdisplay >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nMultipathing Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
multipath -ll >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nNetwork Interfaces:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
ifconfig -a >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nSystem Memory:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
free -m >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nUptime Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
```

Linux (RHEL/CentOS) Patching Process

```
uptime >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nGRUB Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
cat /etc/grub2.cfg >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nProcesses Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
ps -elf >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nResource Utilization & Processes Details:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
top -bn 1 2>&1 1 >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nRouting Table Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
route -n >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nRouting Table Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
cat /etc/fstab >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1
```

Verify the script.
# cat precheck.sh

Apply the execute permission on the script.
# chmod +x precheck.sh

Execute the script.
# ./precheck.sh

Verify the output file.
# cat prechech*

5. Get the details of server what type of stuff running on this server. If the Database is running there, you need engage the DB team and request to them to do pre-check from DB end.

6. Check whether server running in cluster or not. If running as cluster, collect the cluster information with the respective team. e.g. VCS/PCS.

7. In case of bare metal servers, check the console/ILO status.

8. Check with application/database teams if they require to exclude any package like kernel, since the application/database may not have the compatibility with the updated kernel.

9. Check the server backups or snapshot status, it may be VM/physical backups. Involve the backup teams for this task.

10. Submit the change request. It will go to the CAB (Change Advisory Board) for the approval.

11. Once the change got scheduled, we can execute a change as per the change window time.

12. At the change window time you can apply the patches manually or using patch management tools like (RedHat Satellite Server, Katello Foreman, Ansible etc.)

a) Applying the patches manually:

Linux (RHEL/CentOS) Patching Process

Managing (installing/updating/downgrading/removing) only one or few packages at a time. Login to the server which you are going to patch and execute the commands from the root user account or with sudo (if doing it from normal user account).

```
# dnf clean all
# dnf repolist all
# dnf check-update
# dnf list all
# dnf update podman -y
# dnf list installed podman
# rpm -qi podman
# dnf downgrade podman -y
# dnf history
# dnf history info
# dnf history info 1
# dnf history info 2
# dnf list installed zsh
# dnf install -y zsh
# dnf list installed zsh
# dnf history
# dnf history info
# dnf history info 6
# dnf history undo 6
# dnf list installed zsh
# reboot
```

Managing (installing/updating/downgrading/removing) all the packages at a time.

```
# dnf clean all
# dnf repolist all
# dnf check-update
# dnf update -y
# reboot
```

In case if we need to downgrade or undo the changes, we can restore the machine from the backup. Because dnf history undo won't work many times in case of applying all the available patches. You can try below commands for the same but better to restore the machine from the backup in case of undoing the changes. (because that is easy as well as time saving.)

```
# dnf history
# dnf history info
# dnf history info 8
# dnf history undo 8
```

To exclude any package(s) during patching, execute the following command.

```
# dnf update -y --exclude=package1,package2,package3
```

b) Applying the patches automatically:

To apply the patches automatically we need any of the tools like ansible, RedHat Satellite Server etc. in case of large number of machines. We will do it using ansible, pre-check and post-check operations will also be performed automatically.

Let's create the pre-checks and post-checks scripts first on the ansible server.

Pre-checks script:

Linux (RHEL/CentOS) Patching Process

```
# su - john
$ vim precheck.sh

#!/bin/bash

echo -e "\nDate & Time Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
date >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nMounted Filesystems:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
df -Th >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nBlock IDs Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
blkid >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nDisks Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
fdisk -l >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nBlock Storage Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
lsblk >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nVolume Groups Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
vgdisplay >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nLogical Volume Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
lvdisplay >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nMultipathing Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
multipath -ll >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nNetwork Interfaces:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
ifconfig -a >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nSystem Memory:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
free -m >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nUptime Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
uptime >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nGRUB Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
cat /etc/grub2.cfg >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nProcesses Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
ps -elf >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nResource Utilization & Processes Details:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
top -bn 1 2>&1 1 >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nRouting Table Information:" >>/home/john/"precheck_$(date +"%d-%m-%Y").txt"
route -n >>/home/john/"precheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nRouting Table Information:" >>/root/"precheck_$(date +"%d-%m-%Y").txt"
cat /etc/fstab >>/root/"precheck_$(date +"%d-%m-%Y").txt" 2>&1
```

Post-checks script:

```
$ vim postcheck.sh
```

Linux (RHEL/CentOS) Patching Process

```
#!/bin/bash

echo -e "\nDate & Time Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
date >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nMounted Filesystems:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
df -Th >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nBlock IDs Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
blkid >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nDisks Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
fdisk -l >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nBlock Storage Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
lsblk >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nVolume Groups Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
vgdisplay >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nLogical Volume Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
lvdisplay >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nMultipathing Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
multipath -ll >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nNetwork Interfaces:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
ifconfig -a >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nSystem Memory:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
free -m >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nUptime Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
uptime >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nGRUB Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
cat /etc/grub2.cfg >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nProcesses Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
ps -elf >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nResource Utilization & Processes Details:" >>/home/john/"postcheck_$(date +"%d-%m-
%Y").txt"
top -bn 1 2>&1 1 >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nRouting Table Information:" >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt"
route -n >>/home/john/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nRouting Table Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
cat /etc/fstab >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1
```

Let’s create an ansible playbook for patching the managed nodes, to perform the pre and post check operations as well as to fetch the results on the ansible server.

```
$ vim patching.yml
---
- name: Patching Playbook
```

```
  hosts: all
  become: true
  tasks:
    - name: Performing the pre-check operations.
      script: /home/john/precheck.sh
      args:
        creates: /home/john/precheck.sh

    - name: Applying all the available patches.
      dnf:
        name: "*"
        state: latest

    - name: Rebooting the machines.
      reboot:
        reboot_timeout: 6000

    - name: Performing the post-check operations.
      script: /home/john/postcheck.sh
      args:
        creates: /home/john/postcheck.sh

    - name: Finding the pre & post check result files.
      shell: (cd /home/john; find . -maxdepth 1 -type f -iname "*.txt") | cut -d'/' -f2
      register: files_to_fetch

    - name: Fetching the result files.
      fetch:
        src: /home/john/{{ item }}
        dest: /home/john
      with_items: "{{ files_to_fetch.stdout_lines }}"
...
```

Execute the ansible playbook to perform the task.

```
$ ansible-playbook patching.yml
```

13. Collect the server post-patch (post-checks) details and document them in case of manual patching.

You can create a script for the same like below:

# vim postcheck.sh

```
#!/bin/bash

echo -e "\nDate & Time Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
date >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nMounted Filesystems:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
df -Th >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nBlock IDs Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
blkid >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nDisks Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
fdisk -l >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nBlock Storage Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
```

Linux (RHEL/CentOS) Patching Process

```
lsblk >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nVolume Groups Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
vgdisplay >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nLogical Volume Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
lvdisplay >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nMultipathing Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
multipath -ll >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nNetwork Interfaces:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
ifconfig -a >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nSystem Memory:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
free -m >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nUptime Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
uptime >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nGRUB Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
cat /etc/grub2.cfg >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nProcesses Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
ps -elf >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nResource Utilization & Processes Details:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
top -bn 1 2>&1 1 >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nRouting Table Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
route -n >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1

echo -e "\nRouting Table Information:" >>/root/"postcheck_$(date +"%d-%m-%Y").txt"
cat /etc/fstab >>/root/"postcheck_$(date +"%d-%m-%Y").txt" 2>&1
```

```
# chmod +x postcheck.sh

# ./postcheck.sh
```

14. Monitor the server health status. (Use monitoring tools in case of large number of servers).

15. Verify the service status of the App/Resources from the concern teams like database teams, application team etc.

16. Notify the client and the concern teams involved in the change.

17. Close the change request (CR).

You have successfully completed the patching activity.

Thank You