The science behind the report:

Identify and remediate configuration problems more effectively with Red Hat Smart Management

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Identify and remediate configuration problems more effectively with Red Hat Smart Management.

We concluded our hands-on testing on September 23, 2021. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on August 6, 2021 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

During testing, we captured time as well as steps to complete the use cases. Red Hat Smart Management provided administrator time savings on all four use cases that we tested, and reduced administrator steps in three out of four use cases.

Table 1: Time and steps to complete the Vulnerability use case on 100 hosts.

Method	Time (seconds)	Steps
Red Hat Smart Management	2,753	17
Manual	7,006	20
Percentage win	60.7%	15.0%

Table 2: Time and steps to complete the Drift use case on 90 hosts (of 100 total hosts).

Method	Time (seconds)	Steps
Red Hat Smart Management	1,683	60
Manual	16,000	98
Percentage win	89.5%	38.8%

Table 3: Time and steps to complete the Patch use case on 100 hosts.

Method	Time (seconds)	Steps
Red Hat Smart Management	2,437	17
Manual	4,842	21
Percentage win	49.7%	19.0%

Table 4: Time and steps to complete the Policy use case on 100 hosts.

Method	Time (seconds)	Steps
Red Hat Smart Management	494	64
Manual	1,487	25
Percentage win	66.8%	-156.0%

System configuration information

Table 5: Detailed information on the systems we tested.

System configuration information	Dell™ VxRail™ V470F x 4
BIOS name and version	Dell 2.10.5
Non-default BIOS settings	N/A
Operating system name and version/build number	VMware® ESXi™ 7.0.0/15843807
Date of last OS updates/patches applied	11/02/2020
Power management policy	No Power Cap Policy Set, Balanced
Processor	
Number of processors	2
Vendor and model	Intel® Xeon® E5-2698 v4
Core count (per processor)	20
Core frequency (GHz)	2.20
Stepping	B0
Memory module(s)	
Total memory in system (GB)	512
Number of memory modules	16
Vendor and model	Hynix Semiconductor HMA84GR7MFR4N-UH
Size (GB)	32
Type	DDR-4
Speed (MHz)	2,400
Speed running in the server (MHz)	2,400
Storage controller	
Vendor and model	Dell HBA330 Mini
Cache size	0 MB
Firmware version	16.17.00.05
Driver version	17.00.10.00-1vmw.700.1.0.15843807
Local storage	
Number of drives	9
Drive vendor and model	HGST HUSMM1680ASS204 x 3, HGST HUSMR1619ASS204 x 6
Drive size (GB)	745.21 x 3, 1788.50 x 6
Drive information (speed, interface, type)	12Gb SAS, SSD

System configuration information	Dell™ VxRail™ V470F x 4
Network adapter	
Vendor and model	Intel 82599, Intel Corporation Gigabit 4P X20/I350 rNDC
Number and type of ports	2 x 10GbE, 4x1GbE
Driver version	1.7.1.26, 0.1.1.0
Cooling fans	
Vendor and model	Dell embedded
Number of cooling fans	6
Power supplies	
Vendor and model	Dell PWR SPLY, 1100W, RDNT, LTON
Number of power supplies	2
Wattage of each (W)	1,100

How we tested

Run all ssh commands by root. If EC2 and Azure instances do not permit root login, insert a sudo before all commands.

```
This format indicates text to execute via ssh at the command line or to enter in to create a file in a
vi editor.
```

Setting up the VM templates and creating the local VMs

1. In the local vSphere, set up a system named BaseVMTemplate with one CPU, 2500 MB of memory, one 16GB HDD, and a single network adapter.
2. Perform a minimal install of RHEL 7.6. During installation, set up a User account.
3. Open up ports, and install software to communicate with Satellite.

```
firewall-cmd --add-port="80/tcp" --add-port="443/tcp" --add-port="5647/tcp" \
--add-port="8000/tcp" --add-port="8140/tcp" --add-port="9090/tcp" --add-port="53/udp" \
--add-port="53/tcp" --add-port="67/udp" --add-port="69/udp" --add-port="5000/tcp"
firewall-cmd --runtime-to-permanent
firewall-cmd --list-all
subscription-manager clean
```

4. Shut down the BaseVMTemplate.
5. Clone the BaseVMTemplate to create VRedHat001 through VRedHat002.
6. Connect to each new VRedHat*** system, and change the hostname to correspond to the VM name.

```
hostnamectl set-hostname <VM name>
```

Setting up an admin VM

Use this VM to run automation and scripts. Note: We did not use Ansible to automate any manual test cases; these steps are to properly set up Insights and take advantage of automation when setting up our test environment.

1. Create a clone of the BaseVMTemplate called AdminVM.
2. Use ssh to log into system.
3. Change the hostname to AdminVM:

```
hostnamectl set-hostname AdminVM
```

4. Update package list:

```
yum update
```

5. Install Ansible:

```
yum install -y ansible
```

Setting up Red Hat Satellite 6.8

This section uses the following documentation:

- "Installing Red Hat Satellite Server from a Connected Network," available at https://access.redhat.com/documentation/en-us/red_hat_satellite/6.8/html/installing_satellite_server_from_a_connected_network/index
- "Using Cloud Connector to remediate issues across your Red Hat Satellite infrastructure," available at https://access.redhat.com/documentation/en-us/red_hat_insights/2020-10/html/using_cloud_connector_to_remediate_issues_across_your_red_hat_satellite_infrastructure/index.

Execute all CLI commands in this section on the Satellite VM. You can execute browser instructions from any system on the same network as the Satellite VM.

1. In your local vSphere, set up a system named satellite with four CPU, 20,000MB of memory, one 32GB HDD, and one network adapter connected. Perform a minimal install of RHEL 7.6, using custom storage to reduce the storage for / to 23.13 GB and add a 4.67 GB LVM partition for /var. Set up manual IPv4 settings and configure the full hostname before installation.
2. In your DNS server, create a DNS entry for the Satellite machine.
3. Use ssh to log into the admin VM.
4. Log into the Satellite system, and install Satellite:
   a. Log into the system:

```
ssh-agent bash
```

b. Change the hostname as necessary, and verify DNS resolution:

```
sudo hostnamectl --sethostname satellite
ping -c1 localhost
ping -c1 'hostname -f'
```

c. Register to Red Hat subscription management:

```
sudo subscription-manager register
```

d. Attach the Satellite Infrastructure subscription and Red Hat basic subscription:

```
sudo subscription-manager list --all --available --matches 'Red Hat Satellite Infrastructure
Subscription'
sudo subscription-manager attach --pool=<pool_id from previous command>
sudo subscription-manager list --all --available --matches 'Red Hat Enterprise Linux, Self-Support
(128 Sockets, NFR, Partner Only)'
sudo subscription-manager attach --pool=<pool_id from previous command>
```

e. Verify subscriptions:

```
sudo subscription-manager list --consumed
```

f. Configure the Satellite Server repositories:

```
sudo subscription-manager repos --disable "*"
sudo subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-7-server-satellite-6.8-rpms \
--enable=rhel-7-server-satellite-maintenance-6-rpms \
--enable=rhel-server-rhscl-7-rpms \
--enable=rhel-7-server-ansible-2.9-rpms
sudo yum clean all
sudo yum repolist enabled
```

g. Open up the required network ports:

```
sudo firewall-cmd --add-port="443/tcp" --add-port="8140/tcp" --add-port="9090/tcp" --add-port="5000/
tcp" --add-port="80/tcp" \
--add-port="8000/tcp" --add-port="5646/tcp" --add-port="5647/tcp" --add-port="7/tcp" --add-port="7/
udp" --add-port="53/tcp" \
--add-port="53/udp" --add-port="67/udp" --add-port="69/udp" --add-port="22/tcp"
sudo firewall-cmd --runtime-to-permanent
```

h. Install Satellite Server packages, adding a recent hack introduced here, https://community.theforeman.org/t/katello-installation-broken/21374, needed to install python2.

```
sudo yum -y update
sudo yum -y install wget
wget https://download-ib01.fedoraproject.org/pub/epel/7/aarch64/Packages/p/python2-qpid-1.37.0-4.
el7.noarch.rpm
sudo yum -y localinstall python2-qpid-1.37.0-4.el7.noarch.rpm
sudo yum -y install satellite
```

i. Double-check the clock synchronization (it was already active in our satellite):

```
systemctl status chronyd
```

j. Install the SOS package:

```
sudo yum -y install sos
```

k. Run the installer:

```
sudo chmod a+w /etc/foreman-installer/scenarios.d/satellite.yaml
sudo satellite-installer --scenario satellite \
--foreman-initial-admin-username admin \
--foreman-initial-admin-password redhat \
--foreman-proxy-puppetca true \
--foreman-proxy-tftp true \
--enable-foreman-plugin-discovery
```

5. Export and import a subscription manifests named Satellite1. (Export uses instructions from https://access.redhat.com/documentation/en-us/red_hat_subscription_management/1/html/using_red_hat_subscription_management/using_manifests_con.)
    a. In a browser, open https://access.redhat.com.
    b. Log in.
    c. Click Subscriptions.
    d. Click Subscription Allocations.
    e. Click to the right of Simple Content Access to Enable it.
    f. Create a new Subscription Allocation using the following steps:
        i. Click New Subscription Allocation.
        ii. Next to Name, enter Satellite1.
        iii. Next to Type, click the down arrow, and click Satellite 6.8.
        iv. Click Create.
        v. Under Subscriptions, next to Simple content access, move the slider to Enabled.
        vi. Click Export Manifest, and save the file.
    g. Import the Satellite subscription:
        i. In a browser, navigate to http://<satellite ip>.
        ii. Log in as admin, redhat.
        iii. Click Content.
        iv. Click Subscriptions.
        v. Click Import a Manifest.
        vi. Click Choose File.
        vii. Navigate to and select the appropriate manifest file.
        viii. Click Open.
        ix. A Import Manifest progress bar will open, when it completes the subscriptions will appear.
6. Install Insights, and register Satellite with Insights-client for system monitoring:

```
sudo satellite-maintain packages install -y insights-client
sudo insights-client --register
```

7. Enable repositories, and synchronize.
8. In your browser, access http://<satellite ip>.
    a. Log in as admin, redhat.
    b. Click Content.
    c. Click Red Hat Repositories.
    d. To the left of Available, type Satellite Tools 6.8 (for RHEL 7 Server) (RPMs).
    e. Click Search.
    f. Click Red Hat Satellite Tools 6.8 (for RHEL 7 Server) (RPMs).
    g. Next to x86_64, click the + icon to Enable the repository.
    h. To the left of Available, type Red Hat Enterprise Linux 7 Server (RPMs).
    i. Click Search.
    j. Click Red Hat Satellite Tools 6.8 (for RHEL 7 Server) (RPMs).
    k. Next to x86_64 7Server, click the + icon to Enable the repository.
    l. Click Content.
    m. Click Sync Status.
    n. Next to Red Hat Enterprise Linux Server, click the arrow.
    o. Click the Checkbox next to Red Hat Satellite Tools 6.8 for RHEL 7 Server RPMs x86_64.
    p. Click Synchronize Now.
    q. Wait for the Result column to say Synching Complete.
9. Create a content view and activation key:
    a. In your browser, access http://<satellite ip>.
    b. Log in as admin, redhat.
    c. Click Content.
    d. Click Content Views.
    e. Click Create New View.
    f. Under Name, type View1.
    g. Click Save.

   h. Click the boxes next to Red Hat Enterprise Linux 7 Server RPMs x86_64 7Server and Red Hat Satellite Tools 6.8 for RHEL 7 Server RPMs x86_64.
   i. Click Add Repositories.
   j. Click Publish New Version.
   k. Under Description, type Initial View.
   l. Click Save.
   m. Click Content.
   n. Click Activation keys.
   o. Click Create Activation Key.
   p. Under Name, type ActivationKey1.
   q. Next to Library, click the checkbox.
   r. Under Content View, click the down arrow, and select View1.
   s. Click Save.
10. Register all local hosts to Satellite:
   a. Copy /etc/hosts from the satellite server to the admin system ~/allserver.
   b. Add each local host to satellite:
      i. Edit subscribe.sh:

```
subscription-manager remove --all
subscription-manager unregister
subscription-manager clean
rpm -Uvh http://<satellite IP>/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --org="Default_Organization" --activationkey=ActivationKey1
subscription-manager refresh
subscription-manager repos --enable=rhel-7-server-satellite-tools-6.8-rpms
yum repolist
yum install katello-host-tools
```

   c. On satellite, edit the ~/servers file to contain all local servers:

```
grep VRedHat ~/allserver | awk '{ print $1 }'| head -90 > ~/servers
```

   d. On satellite, edit the copyssh.sh file as follows:

```
username=root
readarray -t SERVERS < ~/servers
for HOST in ${SERVERS[@]}; do
ssh-copy-id -i ~foreman-proxy/.ssh/id_rsa_foreman_proxy.pub ${username}@${HOST}
cat subscribe.sh | ssh -i ~foreman-proxy/.ssh/id_rsa_foreman_proxy ${username}@${HOST}
done
```

   e. Make copyssh.sh executable:

```
chmod 0744 copyssh.sh
```

   f. Execute the looping script, answering questions posed with yes or the system password, as appropriate:

```
ssh-agent bash
./copyssh.sh
```

11. Shut down every VRedHat*** system.
12. Create a Snapshot of each VRedHat*** system. (Before performing any use cases, revert each VM back to the snapshot.)
13. Turn on every VRedHat*** system.
14. Configure the VPN in Satellite.
15. Create the cloud-connected systems using Satellite. (Before performing any new use cases, log into each system and manually undo the remediation changes)

Installing and setting up Cloud Connector

Upload your host inventory to Insights, and install Cloud Connector:

```
satellite-installer \
--enable-foreman-plugin-rh-cloud \
--foreman-proxy-plugin-remote-execution-ssh-install-key true
```

1. In a browser, open the Satellite web UI at https://<satellite IP>.
2. Click Configure.
3. Click Inventory Upload.
4. Click Default_Organization.
5. Click Restart, and wait until Exit Code: displays the pid and an exit code of 0.
6. Click Administer.
7. Click Users.
8. Click Create User.
9. Next to Username, enter User1.
10. Under Authorized by, click the down arrow, and click INTERNAL.
11. Next to Password, type Password.
12. Next to verify, type Password.
13. Click Organization.
14. Ensure Default Organization is in the Selected items box.
15. Ensure Default on login is blank.
16. Click Roles.
17. Click the box next to Administrator.
18. Click Submit.
19. Click Hosts.
20. Click All Hosts.
21. Click the Satellite host.
22. Click Schedule Remote Job.
23. Next to Job Category List, click the down arrow, and select Ansible Playbook.
24. Next to Job Template list, click the down arrow, and select Configure Cloud Connector.
25. Next to satellite_user field, type User1.
26. Next to satellite_password , type Password.
27. Click Submit.
28. Wait for the Results to show 100% Success.
29. Make sure the following is at the end of the /etc/systemd/system/receptor@.service file, if it is not, append it:

```
[Install]
WantedBy=multi-user.target
```

30. Check https://cloud.redhat.com/settings/sources to make sure satellite is there.
31. Use a new browser tab to access https://access.redhat.com/management.api
32. Click GENERATE TOKEN.
33. Click Copy.
34. Return to the original browser tab.
35. Click Administer.
36. Click Settings.
37. Click RH Cloud.
38. In the Value column of the Red Hat Cloud token, click the pencil to edit.
39. Highlight the four stars.
40. Press Delete.
41. Right-click, and click Paste.
42. Click the blue checkmark.
43. Click Configure.
44. Click Insights.
45. Click Sync now.
46. Next to Synchronize Automatically, click the blank box to turn the setting ON.
47. Open a new browser window at cloud.redhat.com/settings/groups.

48. Click on the Group you are a member of. (In our case, it was Default access.)
49. Next to the down arrow, click the box to select all the roles.
50. Click Add role.
51. Click OK.
52. Roles that are already inherited will be removed; click the top box to select all the remaining roles.
53. Click Add to group.

Configuring the Admin system for passwordless SSH

1. Set up passwordless ssh in your environment:
    a. Generate an ssh key (when prompted, press enter to select the default destination for the key, and press enter two more times to have an empty passphrase):

    ```
    ssh-keygen
    ```

    b. Create a Servers file with the IP addresses of all the local systems, and a Servers_Azure file with the IP addresses of all the Azure systems.
    c. Copy the key to all non-cloud test systems (when prompted, enter yes or the root password.):

    ```
    username=root
    readarray -t SERVERS < ~/Servers
    for HOST in ${SERVERS[@]}; do ssh-copy-id -i ~/.ssh/id_rsa.pub ${username}@${HOST}; done
    ```

    d. Validate ssh for the cloud systems:
    e. Use scp to copy the AWS key pair to ~/AWSKeyPair.pem.
    f. Validate the pem file setup works:

    ```
    ssh-agent bash
    ssh-add ~/AWSKeyPair.pem
    ```

    g. Use scp to copy the Azure key pair at the Azure console, and save the private key to ~/AzureKey.pem.
    h. Validate the pem file setup works:

    ```
    chmod 0400 AzureKey.pem
    ssh-agent bash
    ssh-add ~/AzureKey.pem
    ```

2. Create a servers file with a list of the host names of all the test systems to maintain:

    ```
    VRedHat001
    VRedHat002
    …
    ```

Testing the use cases

Test the Red Hat Smart Management use case first. Then, unregister the VMs from Satellite and re-register them using the below commands before performing the manual use case.

```
subscription-manager remove --all
subscription-manager unregister
subscription-manager clean
yum remove -y katello-ca-consumer*
subscription-manager register --auto-attach
subscription-manager refresh
```

Testing Policy

Setting up some machines to fail checks

Disable the firewall on VRedHat001 through VRedhat040.

1. Edit Policyprep.sh as follows:

    ```
    systemctl stop firewalld.service
    systemctl disable firewalld
    yum -y install openscap-scanner
    ```

2. Run the updates:

```
cat serverlist | head -40 > PolicyEdit1
username=root
readarray -t SERVERS < ~/PolicyEdit1
for HOST in ${SERVERS[@]}; do cat Policyprep.sh | ssh ${username}@${HOST}; done
```

3. Leave the AWS VMs and Azure VMs as is, they are already heterogeneous.

Manual workflow

1. Connect to admin system with ssh.
2. Log in.
3. Create a directory for Policy reports:

```
mkdir ~/Policy
```

4. Edit the ~/Policy_servers file to contain all local and Azure servers you want to check the firewall and openscap status on:

```
cp serverlist ~/Policy_servers
```

5. Edit the ~/Policy_servers_EC2 file to contain all AWS systems you want check the firewall and openscap status on:

```
cp serverlist_EC2 ~/Policy_servers_EC2
```

6. Edit the Policy.sh file as follows:

```
policyfilename=policy_"$(hostname)"_$(date +"%Y-%m-%d")
touch "/tmp/${policyfilename}"
sudo systemctl status firewalld | grep Active >> "/tmp/${policyfilename}"
openscapinstalled='sudo yum list installed | grep openscap-scanner'
if [ -z "$openscapinstalled" ]; then openscapinstalled="Openscap not installed"; fi
echo $openscapinstalled >> "/tmp/${policyfilename}"
```

7. Edit the ALL-Policy.sh as follows:

```
username=root
readarray -t SERVERS < ~/Policy_servers
for HOST in ${SERVERS[@]}; do
cat Policy.sh | ssh ${username}@${HOST}
mydate='date +"%Y-%m-%d"'
scp ${username}@${HOST}:/tmp/policy_*_$mydate ~/Policy/policy_${HOST}_${mydate}
cat ~/Policy/policy_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/Policy/Policy_
summary.txt
done
username=ec2-user
readarray -t SERVERS < ~/Policy_servers_EC2
for HOST in ${SERVERS[@]}; do
cat Policy.sh | ssh ${username}@${HOST}
mydate='date +"%Y-%m-%d"'
scp ${username}@${HOST}:/tmp/policy_*_$mydate ~/Policy/policy_${HOST}_${mydate}
cat ~/Policy/policy_${HOST}_${mydate} | awk -v h=$HOST '{print h": "$0} ' >> ~/Policy/Policy_
summary.txt
done
```

8. Make ALL-Policy.sh executable:

```
chmod 0744 ALL-Policy.sh
```

9. Execute the looping script:

```
ssh-agent bash
ssh-add ~/foreman*.pem
./ALL-Policy.sh
```

10. View the status of all the systems:

```
more Policy/Policy_summary.txt
ls ~/Policy
```

11. Prepare the fixes.
    a. Save a Fixfirewall.sh script for later that enables your firewall:

    ```
    sudo systemctl start firewalld; sudo systemctl enable firewalld
    ```

    b. Save a Fixscap.sh script for later that installs openscap:

    ```
    sudo yum -y install openscap-scanner
    ```

12. View the status of all the systems before starting remediation:

    ```
    more Policy/Policy_summary.txt
    ls Policy
    ```

13. Copy the ALL-Policy.sh file to ALL-Firewall_fix.sh and ALL-Scap_fix.sh:

    ```
    cp ALL-Policy.sh ALL-Firewall_fix.sh
    cp ALL-Policy.sh ALL-Scap_fix.sh
    ```

14. Replace all references to Policy.sh with Fixfirewall.sh and Fixscap.sh in the looping scripts ALL-Firewall_fix.sh and ALL-Scap_fix.sh:

    ```
    sed -i -e 's+Policy.sh+Fixfirewall.sh+g' ~/ALL-Firewall_fix.sh
    sed -i -e 's+Policy.sh+Fixscap.sh+g' ~/ALL-Scap_fix.sh
    ```

15. Replace all references to Policy_ with Fixfirewall_ and Fixscap_

    ```
    sed -i -e 'g+Policy_+Fixfirewall_+g' ~/ALL-Firewall_fix.sh
    sed -i -e 'g+Policy_+Fixscap_+g' ~/ALL-Scap_fix.sh
    ```

16. Put lists of all the systems to remediate in the appropriate files:

    ```
    grep "inactive" ~/Policy/Policy_summary.txt | grep -v "aws" | awk -F: '{ print $1 }' > ~/
    Fixfirewall_servers | sort --unique
    grep "inactive" ~/Policy/Policy_summary.txt | grep "aws" | awk -F: '{ print $1 }' > ~/Fixfirewall_
    servers_EC2 | sort --unique
    grep "not installed" ~/Policy/Policy_summary.txt | grep -v "aws" | awk -F: '{ print $1 }' > ~/
    Fixscap_servers | sort --unique
    grep "not installed" ~/Policy/Policy_summary.txt | grep "aws" | awk -F: '{ print $1 }' > ~/Fixscap_
    servers_EC2 | sort –unique
    ```

17. Execute the looping scripts:

    ```
    ./ALL-Firewall_fix.sh

    ./ALL-Scap_fix.sh
    ```

Red Hat Smart Management workflow

1. Go to cloud.redhat.com.
2. Log in.
3. On the Red Hat Insights panel, click Policies.
4. From the Policies screen, click Create policy.
5. Give the policy the name “firewalldnotrunning” and description firewalld is not running.
6. Click Next.
7. Enter the condition text `not (facts.enabled_services contains [‘firewalld’] AND facts.running_processes contains [‘firewalld’])`
8. Click Validate Condition.
9. Click Next.
10. Click the Add trigger actions drop-down menu.
11. Click to select the Email option.
12. In the new tab that opened, under Enable email alerts, click Open email preferences.
13. To the right of policies, next to Instant notification, click the box.
14. Click Submit.
15. Close the tab.
16. Click Next.
17. Next to the Policy is disabled text, click the slider to enable this policy.
18. Click Finish.
19. From the Policies screen, click Create policy.
20. Give the policy the name `openscap-scannernotinstalled` and description `openscap-scanner not installed`

21. Click Next.
22. Enter the condition text `not (facts.installed_packages contains [‘openscap-scanner’]).`
23. Click Validate Condition.
24. Click Next.
25. Click the Add trigger actions drop-down menu.
26. Click to select the Email option.
27. Click Next.
28. Next to the Policy is disabled text, click the slider to enable this policy.
29. Click Finish.
30. Connect to the Admin system using ssh.
31. Log in.
32. Trigger a report on each of the systems:
    a. Edit the ~/Insights_servers file to contain all local servers you want to gather a report from:
    ```
    cp serverlist ~/Insights_servers
    ```
    b. Edit the ~/Insights_servers_EC2 file to contain all AWS systems you want to gather a report from:
    ```
    cp serverlist_EC2 ~/Insights_servers_EC2
    ```
    c. Edit the insights.sh file as follows:
    ```
    sudo insights-client
    ```
    d. Edit the ALL-Insights.sh as follows:
    ```
    username=root
    readarray -t SERVERS < ~/Insights_servers
    for HOST in ${SERVERS[@]}; do
    cat insights.sh | ssh ${username}@${HOST}
    done
    username=ec2-user
    readarray -t SERVERS < ~/Insights_servers_EC2
    for HOST in ${SERVERS[@]}; do
    cat insights.sh | ssh ${username}@${HOST}
    done
    ```
33. Make ALL-Policy.sh executable:
    ```
    chmod 0744 ALL-Insights.sh
    ```
34. Execute the looping script, which will put all the policy alerts in your inbox:
    ```
    ssh-agent bash
    ssh-add ~/foreman*.pem
    ./ALL-Insights.sh
    ```
35. Prepare the fixes.
    a. Save a script for later that enables your firewall:
    ```
    sudo systemctl start firewalld; sudo systemctl enable firewalld
    ```
    b. Save a script for later that installs openscap:
    ```
    yum -y install openscap-scanner
    ```
36. Click firewalld not running.
37. Next to filter by system, click the box with an arrow pointing to the right.
38. Click export to CSV.
39. Open the CSV file in Excel.
40. In an empty square, type ”=TEXTJOIN(“ OR “, TRUE, B:B).
41. Right-click the square, and copy.
42. Log into your local satellite in a new browser tab.
43. Click Monitor.
44. In the submenu, click Jobs.
45. On the top right, click Run Job.
46. Leave the Job category as Commands and the Job template as Run Command – SSH Default.

47. Right-click, and paste into Search Query.
48. Next to Resolves to, click Refresh.
49. Next to command, type the contents of your firewall fix script with the lines separated by semicolons.
50. Click Submit.
51. In Insights, click the openscap-scanner not installed policy.
52. Next to Filter by system, click on the box with an arrow pointing to the right.
53. Click export to CSV.
54. Open the CSV file in Excel.
55. In an empty square, type "=TEXTJOIN(" OR ", TRUE, B:B).
56. Right-click the square, and copy.
57. In Satellite, click Run Job.
58. Leave the Job category as Commands and the Job template as Run Command – SSH Default.
59. Right-click, and paste into Search Query
60. Next to Resolves to, click Refresh.
61. Next to command, type the contents of your openscap fix script with the lines separated by semicolons.
62. Click Submit.

Testing Vulnerability (CVE)

Manual workflow

1. Connect to admin system with ssh.
2. Log in.
3. Create a directory for Vulnerability reports:

```
mkdir ~/Vulnerability
```

4. Edit the ~/Vulnernability_servers file to contain all local and Azure servers you want to check the vulnerability on

```
cp serverlist Vulnerability_servers
```

5. Edit the ~/Vulnerability_servers_EC2 file to contain all AWS systems you want to check the vulnerability on

```
cp serverslist_EC2 ~/Vulnerability_servers_EC2
```

6. Edit the Vulnerability.sh file as follows (CVE's we tested are 2020-24489, 2021-27219, and 2021-25217:

```
CVE="<CVE for test>"
hostname='hostname -f'
Vulnerability_filename=Sys-Vulnerability_"${hostname}"_$(date +"%Y-%m-%d")
touch "/tmp/${Vulnerability_filename}"
sudo yum updateinfo --cve "${CVE}" | grep "Important Security notice" >> "/tmp/${Vulnerability_
filename}"
```

7. Edit the ALL-Vulnerability.sh as follows:

```
username=root
readarray -t SERVERS < ~/Vulnerability_servers
for HOST in ${SERVERS[@]}; do
cat Vulnerability.sh | ssh ${username}@${HOST}
mydate='date +"%Y-%m-%d"'
scp ${username}@${HOST}:/tmp/Sys-Vulnerability*$mydate ~/Vulnerability/
Vulnerability_${HOST}_${mydate}
cat ~/Vulnerability/Vulnerability_${HOST}_${mydate}
| awk -v h=$HOST '{print h": "$0} ' >> ~/Vulnerability/Vulnerability_summary.txt
done
username=ec2-user
readarray -t SERVERS < ~/Vulnerability_servers_EC2
for HOST in ${SERVERS[@]}; do
cat Vulnerability.sh | ssh ${username}@${HOST}
mydate='date +"%Y-%m-%d"'
scp ${username}@${HOST}:/tmp/Sys-Vulnerability*$mydate ~/Vulnerability/
Vulnerability_${HOST}_${mydate}
cat ~/Vulnerability/Vulnerability_${HOST}_${mydate}
| awk -v h=$HOST '{print h": "$0} ' >> ~/Vulnerability/Vulnerability_summary.txt
done
```

8. Make ALL-Vulnerability.sh executable:

```
chmod 0744 ALL-Vulnerability.sh
```

9. Execute the looping script:

```
ssh-agent bash
ssh-add ~/foreman*.pem
./ALL-Vulnerability.sh
```

10. View the status of all the systems before starting remediation:

```
more Vulnerability/Vulnerability_summary.txt
ls Vulnerability
```

11. Copy the ALL-Vulnerability.sh file to ALL-Vulnerability_fix.sh:

```
cp ALL-Vulnerability.sh ALL-Vulnerability_fix.sh
```

12. Edit the FixVulnerability.sh file as follows:

```
CVE="<CVE for test>"
hostname='hostname -f'
Vulnerability_filename=Sys-VulnerabilityFix_"$(hostname)"_$(date +"%Y-%m-%d")
touch "/tmp/${Vulnerability_filename}"
sudo yum update -y --cve "${CVE}" >> "/tmp/${Vulnerability_filename}"
```

13. Replace all references to Vulnerability.sh with FixVulnerability.sh in the looping script ALL-Vulnerability_fix.sh:

```
sed -i -e 's+Vulnerability.sh+FixVulnerability.sh+g' ~/ALL-Vulnerability_fix.sh
```

14. Replace all references to Vulnerability_ with VulnerabilityFix_

```
sed -i -e 'g+Vulnerability_+VulnerabilityFix_+g' ~/ALL-Vulnerability_fix.sh
```

15. Put lists of all the systems to remediate in the appropriate files:

```
grep "Important Security notice(s)" ~/Vulnerability/Vulnerability_summary.txt | grep -v "aws" | awk
-F: '{ print $1 }' > ~/VulnerabilityFix_servers
grep "Important Security notice(s)" ~/Vulnerability/Vulnerability_summary.txt | grep "aws" | awk
-F: '{ print $1 }' > ~/VulnerabilityFix_servers_EC2
```

16. Execute the looping script:

```
./ALL-Vulnerability_fix.sh
```

Red Hat Smart Management workflow

1. Go to cloud.redhat.com.
2. Log in.
3. On the Red Hat Insights panel, click Vulnerability.
4. Click the Systems tab.
5. Click on the first system.
6. Notice that there is a CVE-2020-8616.
7. Gather information on the CVE.
    a. Click the CVE.
    b. Read through the text.
    c. Scroll down to see a list of exposed systems.
    d. Underneath Exposed systems, to the left of the down arrow, click the box to select all systems.
    e. Click the blue Remediate button.
    f. Next to Create new playbook, type CVE-2020-8616.
    g. Click Next.
    h. Click Create.
    i. Click Remediations.
    j. Click CVE-2020-8616.
    k. Click Execute playbook.

Testing Drift (baseline and change)

Use one system as the baseline VM to compare the other VMs to. To do the most comparisons while still have the comparisons make sense, we recommend using local VM VRedHat001for the baseline, and then making changes to each of the 90 local VMs. These changes should be detected as drift.

Manual workflow – Creating a baseline

1. Connect to admin system with ssh.
2. Log in.
3. Edit the Createbaseline.sh file:

```
baselinefilename=baseline_"$(hostname)"_$(date +"%Y-%m-%d")
rm -f "/tmp/${baselinefilename}"
bios='dmidecode -s bios-version'
if [[ $bios = *mazon* ]]; then echo cloud_provider=Amazon >> "/tmp/${baselinefilename}"
elif [[ $bios = *zure* ]]; then echo cloud_provider=Azure >> "/tmp/${baselinefilename}"
else echo cloud_provider=N/A >> "/tmp/${baselinefilename}"; fi
echo arch='uname -m' >> "/tmp/${baselinefilename}"
echo bios_release_date='dmidecode -s bios-release-date' >> "/tmp/${baselinefilename}"
echo bios_vendor='dmidecode -s system-manufacturer' >> "/tmp/${baselinefilename}"
echo bios_version='dmidecode -s bios-version' >> "/tmp/${baselinefilename}"
echo cpu_info='lscpu' >> "/tmp/${baselinefilename}"
echo enabled_services='systemctl list-unit-files' >> "/tmp/${baselinefilename}"
echo infrastructure_type='dmidecode -s system-product-name' >> "/tmp/${baselinefilename}"
echo infrastructure_vendor='dmidecode -s system-manufacturer' >> "/tmp/${baselinefilename}"
echo installed_packages='rpm -qa --last' >> "/tmp/${baselinefilename}"
echo kernel_modules='lsmod' >> "/tmp/${baselinefilename}"
echo os_kernel_version='uname -r' >> "/tmp/${baselinefilename}"
echo os_release='cat /etc/redhat-release' >> "/tmp/${baselinefilename}"
echo running_processes='ps aux | awk '{print $11}'' >> "/tmp/${baselinefilename}"
echo subscriptions='subscription-manager status' >> "/tmp/${baselinefilename}"
echo system_memory='free -h | grep Mem | awk '{print $2}'' >> "/tmp/${baselinefilename}"
echo yum_repos='yum repolist' >> "/tmp/${baselinefilename}"
ssh root@<admin server> 'if [ ! -f "~/Drift" ]; then mkdir ~/Drift; fi'
scp "/tmp/${baselinefilename}" "root@<admin system ip>:~/Drift/${baselinefilename}"
```

4. Execute Createbaseline.sh

```
cat ./Createbaseline.sh | ssh root@vredhat001.bannister.local
```

Manual workflow - Making some changes to the systems

1. Run updates on VRedHat001 through VRedHat030.
   a. ssh to the admin system.
   b. Log in.
   c. Edit Patchprep1.sh as follows:

   ```
   sudo yum update -y
   ```

   d. Run the updates:

   ```
   cat serverlist | head -30 > DriftEdit1
   username=root
   readarray -t SERVERS < ~/PatchEdit1
   for HOST in ${SERVERS[@]}; do cat Patchprep1.sh | ssh ${username}@${HOST}; done
   ```

2. Install extra packages on VRedHat031 through VRedHat060.
   a. Edit Patchprep2.sh as follows:

   ```
   sudo yum install python3 -y
   ```

   b. Run the updates:

   ```
   cat serverlist | head -60 | tail -n 30 > DriftEdit2
   username=root
   readarray -t SERVERS < ~/PatchEdit2
   for HOST in ${SERVERS[@]}; do cat Patchprep2.sh | ssh ${username}@${HOST}; done
   ```

3. Load a new kernel module on VRedHat061 through VRedHat090.
   a. Edit Patchprep3.sh to look like this:

```
modprobe wacom
```

   b. Run the updates:

```
cat serverlist | head -90 | tail -n 30 > DriftEdit3
username=root
readarray -t SERVERS < ~/PatchEdit3
for HOST in ${SERVERS[@]}; do cat Patchprep3.sh | ssh ${username}@${HOST}; done
```

Manual workflow: Comparing the configuration against a baseline and performing remediation

1. Connect to admin system with ssh.
2. Log in.
3. Create a directory for Drift reports, if necessary:

```
if [ ! -f "~/Drift" ]; then mkdir ~/Drift; fi
```

4. Edit the ~/Drift_servers file to contain all local and Azure servers you want to check the vulnerability on:

```
cp serverlist ~/Drift_servers
```

5. Copy the Createbaseline.sh to the admin system:

```
scp root@vredhat001.bannister.local:~/Createbaseline.sh ~/Drift.sh
```

6. Edit Drift.sh:

```
sed -i -e 's+baselinefilename+myfilename+g' ~/Drift.sh
sed -i -e 's+myfilename=baseline_+myfilename=driftcompare_+g' ~/Drift.sh
sed -i -e 's/ssh/d' ~/Drift.sh
sed -i -e 's/scp/d' ~/Drift.sh
```

7. Edit the ALL-Drift.sh to look like this so it will compare all systems to the baseline we created earlier:

```
username=root
readarray -t SERVERS < ~/Drift_servers
for HOST in ${SERVERS[@]}; do
cat Drift.sh | ssh ${username}@${HOST}
mydate='date +"%Y-%m-%d"'
scp ${username}@${HOST}:/tmp/driftcompare* ~/Drift/driftcompare_${HOST}_${mydate}
touch ~/Drift/Drift_summary.txt
diff Drift/driftcompare_${HOST}_${mydate} Drift/baseline* | awk -v h=$HOST '{print h": "$0} ' >> ~/
Drift/Drift_summary.txt
done
```

8. Make ALL-Drift.sh executable:

```
chmod 0744 ALL-Drift.sh
```

9. Execute the looping script:

```
./ALL-Drift.sh
```

10. View the status of all the systems, to decide what to do to remediate each one:

```
cat Drift/Drift_summary.txt
```

   - When we notice systems that have been updated from the baseline configuration, the appropriate remediation is to create a new baseline with the updates and bring the rest of the systems up to the new baseline.
   - When we notice systems that have python3 installed, we might decide that we need to remove the package from our systems.
   - When we notice systems that have a wacom module running, which we do not need, we need to stop the module and prevent it from starting at reboot if necessary.

11. For each remediation step that needs to occur, give it a number and create 2 files:
   - RemediationX.sh: Contains the commands to return the system(s) to baseline or bring the system up to the new baseline.
   - ListX: Contains the systems that need this remediation.

12. Update any systems that need it and create a new Drift_summary file:
   a. Edit Remediation1.sh

```
sudo yum update -y
```

b. Find servers that need the OS updating:

```
grep os_release= baseline* > Baselineos
grep os_release= Drift_summary.txt > Driftos
input=Driftos
while IFS= read -r line; do echo "$line" > tmp; host='awk -F: '{ print $1 }' tmp'; echo $host >>
alreadyupdated; sort alreadyupdated > alreadyupdatedS; done < "$input"
sort serverlist > tmplist
comm -1 -3 alreadyupdatedS tmplist | awk -F" " '{ print $1 }' > List1
```

c. Perform update and create new Baseline and Drift_summary:

```
cp Drift.sh Drift1.sh
sed -i -e 's+driftcompare_+driftcompare1_+g' Drift1.sh
cp ALL-Drift.sh ALL-Drift1.sh
sed -i -e 's+Drift.sh+Drift1.sh+g' ALL-Drift1.sh
sed -i -e 's+driftcompare_+driftcompare1_+g' ALL-Drift1.sh
username=root
readarray -t SERVERS < ~/List1
for HOST in ${SERVERS[@]}; do
    cat Remediation1.sh | ssh ${username}@${HOST}
done
       cat ./createbaseline.sh | ssh root@vredhat001.bannister.local
./ALL-Drift1.sh
```

13. Check if any systems have additional enabled services, find none.

```
grep enabled_services baseline* > Baselineservices
sed -i -e 's+ +\n+g' Baselineservices
grep enabled_services Drift_summary.txt > Driftservices
sort -u Baselineservices > BaselineservicesS
input=Driftservices
while IFS= read -r line; do echo "$line" > tmp; sed -i -e 's+ +\n+g' tmp; sed 's+:++g' <<<$(head -n
1 tmp) >> servicesdiff; sort -u tmp > tmpS; comm -3 tmpS BaselineservicesS >> servicesdiff; done <
"$input"
grep -v "<\|>\|:\|scope" servicesdiff | more
```

14. Check if any systems have additional installed packages, find python3.

a. Find out that python3 has been installed and find servers that need python3 removed.

```
grep installed_packages baseline* > Baselinepackages
sed -i -e 's+ +\n+g' Baselinepackages
grep installed_packages Drift_summary.txt > Driftpackages
sort -u Baselinepackages > BaselinepackagesS
input=Driftpackages
while IFS= read -r line; do echo "$line" > tmp; sed -i -e 's+ +\n+g' tmp; sed 's+:++g' <<<$(head -n
1 tmp) >> packagesdiff; sort -u tmp > tmpS; comm -3 tmpS BaselinepackagesS >> packagesdiff; done <
"$input"
       grep x86_64 packagesdiff | sort | uniq | more
grep -v "<\|>\|:\|[a-z]" packagesdiff | grep bannister.local | uniq > List2
```

b. Edit Remediation2.sh:

```
sudo yum remove python3 -y
```

c. Perform python removal and create new Baseline and Drift_summary:

```
cp Drift.sh Drift2.sh
sed -i -e 's+driftcompare_+driftcompare2_+g' Drift2.sh
cp ALL-Drift.sh ALL-Drift2.sh
sed -i -e 's+Drift.sh+Drift2.sh+g' ALL-Drift2.sh
sed -i -e 's+driftcompare_+driftcompare2_+g' ALL-Drift2.sh
username=root
readarray -t SERVERS < ~/List2
for HOST in ${SERVERS[@]}; do
   cat Remediation2.sh | ssh ${username}@${HOST}
done
cat ./createbaseline.sh | ssh root@vredhat001.bannister.local
./ALL-Drift2.sh
```

15. Check if any systems have additional kernel modules running, find wacom.
16. Find out that the Wacom module has been loaded and find servers that need it removed.

```
grep kernel_modules baseline* > Baselinemoduleswjunk
sed -i -e 's+ +\n+g' Baselinemoduleswjunk
grep "[a-z]" Baselinemoduleswjunk > Baselinemodules
grep kernel_modules Drift_summary.txt > Driftmodules
sort -u Baselinemodules > BaselinemodulesS
input=Driftmodules
while IFS= read -r line; do echo "$line" > tmpwjunk; sed -i -e 's+ +\n+g' tmpwjunk; sed 's+:++g'
<<<$(head -n 1 tmpwjunk) >> modulesdiff; grep "[a-z]" tmpwjunk > tmp; sort -u tmp > tmpS; comm
-3 tmpS BaselinemodulesS >> diffwspaces; sed -i -e 's+ ++g' diffwspaces >> modulesdiff; done <
"$input"
       grep -v "<\|>\|:" modulesdiff | grep bannister.local | uniq > List3
```

17. Edit Remediation2.sh:

```
modprobe -r wacom
sed -i -e 's/wacom/d' /etc/modules-load.d/*.conf
sed -i -e 's/wacom/d' /etc/modules
```

18. Perform wacom unload and create new Baseline and Drift_summary:

```
cp Drift.sh Drift2.sh
sed -i -e 's+driftcompare_+driftcompare3_+g' Drift3.sh
cp ALL-Drift.sh ALL-Drift3.sh
sed -i -e 's+Drift.sh+Drift3.sh+g' ALL-Drift3.sh
sed -i -e 's+driftcompare_+driftcompare3_+g' ALL-Drift3.sh
username=root
readarray -t SERVERS < ~/List3
for HOST in ${SERVERS[@]}; do
   cat Remediation3.sh | ssh ${username}@${HOST}
done
cat ./createbaseline.sh | ssh root@vredhat001.bannister.local
./ALL-Drift3.sh
```

Red Hat Smart Management workflow – Creating a baseline

1. Go to cloud.redhat.com.
2. Log in.
3. On the Red Hat Insights panel, click Drift.
4. On the left side of the screen, under Drift, click Baselines.
5. Click Create baseline.
6. To the left of Copy an existing system, click the radio button.
7. Underneath the Baseline name text, type the hostname and date in the textbox.
8. To the left of the name of the system desired, click the checkbox to select it as the baseline system.
9. Click Create baseline.
10. To the right of last_boot_time, click the column of three dots, and click Delete fact.
11. To the right of fqdn, click the column of three dots, and click Delete fact.
12. Click Delete facts.
13. To the left of network interfaces, click the column of three dots, and click Delete category.

Red Hat Smart Management workflow – Making some changes to the systems

The same changes should be made to the same systems as we did in the manual use case.

Red Hat Smart Management workflow – Comparing the current configurations against a baseline and performing remediation

1. Manually start a scan on all the systems (In a production environment each system will be scanned nightly and all drift information will already be in insights the morning after the change occurred, without user intervention. For this reason, the manual scan steps and times will not be included in our analysis):
    a. Connect to the Admin system with ssh.
    b. Log in.
    c. Edit the ~/Drift_Scan_servers file to contain all local servers you want to check the Drift on:

```
grep VRedHat ~/allserver | awk '{ print $1 }' | head -90 > ~/Drift_Scan_servers
```

d. Edit the Drift_Scan.sh file as follows:

```
insights-client
```

e. Edit the ALL_Driftscan.sh as follows:

```
username=root
readarray -t SERVERS < ~/Drift_Scan_servers
for HOST in ${SERVERS[@]}; do
cat Drift_Scan.sh | ssh ${username}@${HOST}
done
```

f. Make ALL-DriftScan.sh executable:

```
chmod 0744 ALL-DriftScan.sh
```

g. Execute the looping script to trigger an update in Insights for all the systems:

```
ssh-agent bash
./ALL-DriftScan.sh
```

2. Go to cloud.redhat.com.
3. Log in.
4. Click All apps and services.
5. Click Drift.
6. Click Add to comparison.
7. Beside 1-50 of 101, click the down arrow, and select 100 per page.
8. To the right of the Name column header, enter vredhat to eliminate the systems we don't want to compare to.
9. To the left of the Name column header, click the box to select all systems. Scroll to see the correct systems are checked off.
10. Click the Baselines tab.
11. To the left of the name of the baselines desired, click the checkbox to select it as the baseline to compare to.
12. Click Submit.
13. Scroll to the right to see all the systems with the wrong OS release. Make a note of it.
14. Click the > next to installed packages and scroll down to see that python3 is installed on some systems that it shouldn't be.
15. Scroll up to see the list of systems, scroll to the right to bring the next group of 3-4 systems onto the screen, and then scroll back down to python3 repeatedly to see if the current group of systems has python3 installed. Make a note of it.
16. Repeat step 15 until we have viewed all systems and know which systems have python3 installed.
17. Click the down arrow next to installed_packages and click the > next to kernel modules. See that wacom is installed on some systems that it shouldn't be.
18. Scroll to the right to see all the systems with the wacom module installed. Make a note of it.
19. Go to a new tab with https://<your satellite IP>/users/login.
20. Log in.
21. On the left, click Monitor.
22. On the menu that appears, click Jobs.
23. On the right, click Run Job.
24. Next to Search Query, enter "vredhat and (t031 or t032 or t033 or t034 or t035 or t036 or t037 or t038 or t039 or t04 or t05 or t06 or t07 or t08 or t09)"
25. Below the Search Query textbox, click the refresh button and note that this resolves to 60 hosts.
26. Click on the eye button to preview the list of hosts and scroll to make sure in includes vredhat031 through vredhat090. (Which are the systems that you made notes on to indicate they had the wrong OS)
27. Close the preview window
28. In the command box, type yum update -y to update the systems that still have the old OS release
29. Click Submit.
30. Click Monitor.
31. On the menu that appears, click Jobs.
32. On the right, click Run Job.
33. Next to Search Query, type vredhat and (t031 or t032 or t033 or t034 or t035 or t036 or t037 or t038 or t039 or t060 or t04 or t05.
34. Below the Search Query textbox, click Refresh and see that it resolves to 30 hosts.
35. Click the eye n to preview the list of hosts and scroll to make sure it includes vredhat031 through vredhat060 (i.e., the systems that you made notes on to indicate they had pyhon3 packages).
36. Close the preview window.
37. In the command box, type yum remove python3 -y.
38. Click Submit.

39. Click Monitor.
40. On the menu that appears, click Jobs.
41. On the right, click Run jobs.
42. Next to Search Query, type vredhat and (t061 or t062 or t063 or t064 or t065 or t066 or t067 or t068 or t069 or t07 or t08j or t09).
43. Below the Search Query textbox, click Refresh, and see that it resolves to 30 hosts
44. Click the eye to preview the list of hosts, and scroll to make sure it includes vredhat061 through vredhat090 (i.e., the systems that you made notes on to indicate they had the wacom module loaded).
45. Close the preview window.
46. In the command box, type modprobe -r wacom; echo “blacklist wacom” >> /etc/modprobe.d/local-dontload.conf; echo “install wacom /bin/false” >> /etc/modprobe.d/local-dontload.conf.
47. Wait for the 100% Success to appear, indicating the wacom job is finished.
48. Click Monitor.
49. On the menu that appears, click Jobs.
50. View the list of running jobs, and wait until all three jobs show a status of Succeeded.

Testing Patch (patches)

Manual workflow

1. Connect to admin system with ssh.
2. Log in.
3. Create a directory for Patch reports:

```
mkdir ~/Patch
```

4. Edit the ~/Patch_servers file to contain all local and Azure servers you want to check the vulnerability on

```
cp serverlist ~/Patch_servers
```

5. Edit the ~/Patch_servers_EC2 file to contain all AWS systems you want to check the vulnerability on

```
cp serverlist_EC2 ~/Patch_servers_EC2
```

6. Edit the Patch.sh file as follows (Patches we tested are RHBA-2021:2320, RHSA-2021:2314, RHBA-2021:2312):

```
PATCH="<PATCH for test>"
hostname='hostname -f'
Patch_filename=SysPatch_"${hostname}"_$(date +"%Y-%m-%d")
touch "/tmp/${Patch_filename}"
sudo yum updateinfo --advisory "${PATCH}" | grep "notice(s)" >> "/tmp/${Patch_filename}"
```

7. Edit the ALL-Patch.sh as follows:

```
username=root
readarray -t SERVERS < ~/Patch_servers
for HOST in ${SERVERS[@]}; do
cat Patch.sh | ssh ${username}@${HOST}
mydate='date +"%Y-%m-%d"'
scp ${username}@${HOST}:/tmp/SysPatch*$mydate ~/Patch/Patch_${HOST}_${mydate}
cat ~/Patch/Patch_${HOST}_${mydate}
| awk -v h=$HOST '{print h": "$0} ' >> ~/Patch/Patch_summary.txt
done
username=ec2-user
readarray -t SERVERS < ~/Patch_servers_EC2
for HOST in ${SERVERS[@]}; do
cat Patch.sh | ssh ${username}@${HOST}
mydate='date +"%Y-%m-%d"'
scp ${username}@${HOST}:/tmp/SysPatch*$mydate ~/Patch/Patch_${HOST}_${mydate}
cat ~/Patch/Patch_${HOST}_${mydate}
| awk -v h=$HOST '{print h": "$0} ' >> ~/Patch/Patch_summary.txt
done
```

8. Make ALL-Patch.sh executable:

```
chmod 0744 ALL-Patch.sh
```

9. Execute the looping script:

```
ssh-agent bash
ssh-add ~/foreman*.pem
./ALL-Patch.sh
```

10. View the status of all the systems before starting remediation:

```
more Patch/Patch_summary.txt
ls Patch
```

11. Copy the ALL-Patch.sh file to ALL-Patch_fix.sh:

```
cp ALL-Patch.sh ALL-Patch_fix.sh
```

12. Edit the FixPatch.sh file as follows:

```
PATCH="<PATCH for test>"
hostname='hostname -f'
Patch_filename=SysPatchFix_"${hostname}"_$(date +"%Y-%m-%d")
touch "/tmp/${Patch_filename}"
sudo yum update -y --advisory "${PATCH}" >> "/tmp/${Patch_filename}"
```

13. Replace all references to Patch.sh with FixPatch.sh in the looping script ALL-Patch_fix.sh:

```
sed -i -e 's+Patch.sh+FixPatch.sh+g' ~/ALL-Patch_fix.sh
```

14. Replace all references to Patch_ with PatchFix_

```
sed -i -e 'g+Patch_+PatchFix_+g' ~/ALL-Patch_fix.sh
```

15. Put lists of all the systems to remediate in the appropriate files:

```
grep "notice(s)" ~/Patch/Patch_summary.txt | grep -v "aws" | awk -F: '{ print $1 }' > ~/PatchFix_
servers
grep "notice(s)" ~/Patch/Patch_summary.txt | grep "aws" | awk -F: '{ print $1 }' > ~/PatchFix_
servers_EC2
```

16. Execute the looping script:

```
./ALL-Patch_fix.sh
```

Red Hat Smart Management workflow

1. Go to cloud.redhat.com.
2. Log in.
3. On the Red Hat Insights panel, click Patch.
4. Notice that there is a RHBA-2020:2355.
5. Gather information on the bugfix:
    a. Click on the bugfix.
    b. Read through the text.
    c. Scroll down to see a list of Affected systems.
    d. Underneath affected systems, to the left of the down arrow, click the box to select all systems.
    e. Click Remediate.
    f. Next to Create new playbook, type Patch.
    g. Click Next.
    h. Click Create.
    i. Click Remediations.
    j. Click Patch.
    k. Click Execute playbook.

Read the report at https://facts.pt/s3fskyn ▶

This project was commissioned by Red Hat.

Principled Technologies®

Facts matter.®