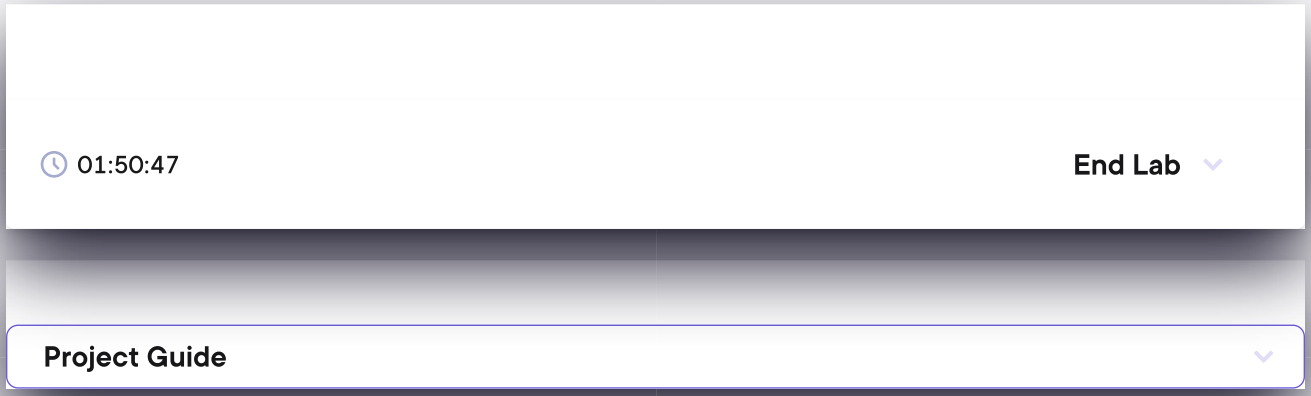
Red Hat Certified Specialist in Linux Diagnostics and Troubleshooting (EX342) Exam

By Elle Krout



Red Hat Certified Specialist in Linux Diagnostics and Troubleshooting (EX342)

Description

The Red Hat Certified Specialist in Linux Diagnostics and Troubleshooting exam tests on a person's ability to resolve issues with broken systems, at variously levels of the system. This includes system startup, file and package management, authentication, networking, and the ability to leverage system tools to gather data about each of these topics. In this hands-on practice exam, we'll practice diagnosing and troubleshooting four provided servers with a myriad of problems.

This practice exam is not approved or sponsored by Red Hat.

Solution

> **Note**: All servers must have `firewalld` and SELinux enabled and running. Issues cannot be resolved by turning off security features.

1. Open four terminal windows.
2. Connect to `HostA`, `HostB`, `HostC`, and `HostD` using the credentials prov

   ```
   ssh cloud_user@<PUBLIC_IP_ADDRESS>
   ```

3. On each server, become the root user, entering the provided password when prompted:

   ```
   sudo -i
   ```

Tasks 1–3

1. Upgrade a Package

1. On `HostA`, upgrade the `httpd` package to the latest release:

   ```
   [root@HostA ~]# dnf upgrade httpd -y
   ```

   Observe the note indicating the dependencies are resolved.
2. Check to see if `httpd` is versionlocked:

   ```
   [root@HostA ~]# dnf versionlock list
   ```

   `httpd` is indeed versionlocked.
3. Unlock the package:

   ```
   [root@HostA ~]# dnf versionlock delete httpd
   ```
4. Try upgrading `httpd` again:

   ```
   [root@HostA ~]# dnf upgrade httpd -y
   ```

   This time it should be successful.
5. Relock the package:

   ```
   [root@HostA ~]# dnf versionlock add httpd
   ```

2. Examine Application

1. On `HostA`, start by moving to `/opt/`:

   ```
   [root@HostA ~]# cd /opt/
   ```
2. Review the files within the directory (`ls`). Observe the `yaga_app` is there.
3. Check if `valgrind` is installed:

   ```
   [root@HostA ~]# valgrind -v
   ```

   It is not found.
4. Install `valgrind`:

   ```
   [root@HostA ~]# dnf install valgrind -y
   ```
5. Check for memory issues with `yaga_app`:

   ```
   [root@HostA ~]# valgrind ./yaga_app
   ```
6. From the output, you will need to re-run the command with the `--leak-check=full` and `--show-leak-kinds=all` flags:

   ```
   [root@HostA ~]# valgrind --leak-check=full --show-leak-kinds=all ./yaga_app
   ```

The output indicates there are some bytes that are still reachable.

7. Create a report for these issues, and save it for the app maintainers:

```
[root@HostA ~]# valgrind --leak-check=full --show-leak-kinds=all --log-file=/opt/reports/yaga_memcheck ./yaga_app
```

8. Confirm the report was saved:

```
[root@HostA ~]# ls reports/
```

3. Uncover the Source of CPU Spikes

1. On `HostD`, review the current CPU usage:

```
[root@HostD ~]# pcp atop
```

Observe the three processes at the top of the second list that are using a lot of CPU. Note the `PID` in the first column for these processes. Remember these values for the next step. Press **CTRL + C** to exit.

2. End the three processes by running the following command three times. Replace `<Process_ID>` with the `PIDs` you just retrieved:

```
[root@HostD ~]# kill -9 <PROCESS_ID>
```

3. Confirm the processes are no longer active:

```
[root@HostD ~]# pcp atop
```

You should no longer see those processes.

Tasks 4–6

4. Resolve Remote Logging Issues

For this task, you will work in both `HostA` and `HostB`. `HostB` is set up to accept remote logging from other servers, but it is not receiving logs from `HostA`.

1. On `HostB`, review the `rsyslog.conf`:

```
[root@HostB ~]# cat /etc/rsyslog.conf
```

Observe you are using the default port of `514`.

2. Check that `rsyslog` is running:

```
[root@HostB ~]# systemctl status rsyslog
```

Everything seems to be functioning properly. Press **CTRL + C** to exit.

3. Check the firewall on `HostB`:

```
[root@HostB ~]# firewall-cmd --list-all
```

There doesn't seem to be any ports open for `rsyslog`.

4. Add port `514`:

```
[root@HostB ~]# firewall-cmd --permanent --add-port=514/tcp
```


5. Reload the firewall rules:

   ```
   [root@HostB ~]# firewall-cmd --reload
   ```

6. Restart `rsyslog`:

   ```
   [root@HostB ~]# systemctl restart rsyslog
   ```

7. On `HostA`, check the current status of `rsyslog`:

   ```
   [root@HostA ~]# systemctl status rsyslog
   ```

   Observe it was previously refusing to connect to the port.

8. Restart the service:

   ```
   [root@HostA ~]# systemctl restart rsyslog
   ```

9. Recheck for errors:

   ```
   [root@HostA ~]# systemctl status rsyslog
   ```

   There doesn't seem to be any errors about port connection.

10. Attempt to send a log to `HostB`:

    ```
    [root@HostA ~]# logger test -p user.emerg
    ```

    You should see a response on both `HostA` and `HostB`.

11. Review the logs from `HostA` on `HostB`:

    ```
    [root@HostB ~]# tail /var/log/hosts/HostA.log
    ```

    You should see the `test` log as the last one listed.

5. Resolve Package Manager Problem

When trying to download a new package using `dnf` on `HostB`, the download fails.

1. Replicate the issue by trying to install `vim`:

   ```
   [root@HostB ~]# dnf install vim -y
   ```

   You should see a number of errors, specifically indicating that it cannot open the `Packages` index.

2. Move to the `rpm` directory:

   ```
   [root@HostB ~]# cd /var/lib/rpm
   ```

3. Verify the `Packages` file:

   ```
   [root@HostB ~]# /usr/lib/rpm/rpmdb_verify Packages
   ```

   It will fail verification.

4. Check for open package locks:

   ```
   [root@HostB ~]# rm -rf /var/lib/rpm/__db*
   ```

5. Move the current `Packages` file to a backup `Packages` file:

   ```
   [root@HostB ~]# mv Packages Packages.bak
   ```

6. Dump the backed up file to a new `Packages` database:

```
[root@HostB ~]# /usr/lib/rpm/rpmdb_dump Packages.bak | /usr/
lib/rpm/rpmdb_load Packages
```

7. Verify the `Packages` file:

   ```
   [root@HostB ~]# /usr/lib/rpm/rpmdb_verify Packages
   ```
   This time it is successful.

8. Rebuild the database:

   ```
   [root@HostB ~]# rpm --rebuilddb
   ```

9. Reattempt to install `vim`:

   ```
   [root@HostB ~]# dnf install vim -y
   ```
   This time another error appears due to the release version.

10. `echo` the version you are on to `/etc/yum/vars/releasever`:

    ```
    [root@HostB ~]# echo "8" > /etc/yum/vars/releasever
    ```

11. Attempt to install `vim` one more time. This time you should be successful.

6. Determine the Issue with Performance Co-Pilot

1. On `HostB`, attempt to start the `pmcd` service:

   ```
   [root@HostB ~]# systemctl start pmcd
   ```
   You should see the error `Transaction order is cyclic.`

2. Attempt to start `pmlogger`:

   ```
   [root@HostB ~]# systemctl start pmlogger
   ```
   You should see the same error. This means that there are most likely conflicting dependencies. Press **CTRL + C** to exit.

3. Check the status for `pmcd`:

   ```
   [root@HostB ~]# systemctl status pmcd
   ```
   Observe it refers to a `dependency.conf` file as the `Drop-In`.

4. Check the status of `pmlogger`:

   ```
   [root@HostB ~]# systemctl status pmlogger
   ```
   You should see a `Drop-In` file for this service in its own location

5. Review the `pmlogger` and `pmcd` files:

   ```
   [root@HostB ~]# cat /etc/systemd/system/pmlogger.service.d/
   dependency.conf
   [root@HostB ~]# cat /etc/systemd/system/pmcd.service.d/
   dependency.conf
   ```
   Observe both files are similar and have the same issues and have conflicting dependencies.

6. Remove the `pmcd` dependency:

```
[root@HostB ~]# rm /etc/systemd/system/pmcd.service.d/
dependency.conf
```
When prompted, type `y` to confirm.

7. Try starting `pmlogger` again:

   ```
   [root@HostB ~]# systemctl start pmlogger
   ```
   You should see the same error — but this is because you still need to complete another step.

8. Run a `daemon-reload`:

   ```
   [root@HostB ~]# systemctl daemon-reload
   ```

9. Try starting `pmlogger` one more time:

   ```
   [root@HostB ~]# systemctl start pmlogger
   ```
   This time is should start, and since `pmcd` has the dependency, it should start as well.

10. Verify by checking the status of `pmcd`:

    ```
    [root@HostB ~]# systemctl status pmcd
    ```
    Everything should be active and running.

Tasks 7–9

7. Repair Corrupted File System

1. On `HostC`, review `fstab`:

   ```
   [root@HostC ~]# cat /etc/fstab
   ```
   You should see an `/images` mount point currently set up.

2. Try to mount against `/images`:

   ```
   [root@HostC ~]# mount /images/
   ```
   You will run into an error indicating the file system is corrupted.

3. Repair the file system:

   ```
   [root@HostC ~]# xfs_repair /dev/nvme1n1p1
   ```

4. Try to mount against `/images` again:

   ```
   [root@HostC ~]# mount /images/
   ```

5. Check the `/images` directory to verify:

   ```
   [root@HostC ~]# ls /images/
   ```
   You should see the files in the directory.

8. Restore to an LVM Archive and Recover Encrypted Data

1. On `HostD`, attempt to mount `/data/`:

```
[root@HostD ~]# mount /data/
```

Observe the special device at `/dev/mapper` does not exist.

2. Review `fstab`:

```
[root@HostD ~]# cat /etc/fstab
```

Observe that the special device mentioned when running `mount` is also mentioned here, but there isn't a lot of other useful information.

3. Review `/etc/crypttab`:

```
[root@HostD ~]# cat /etc/crypttab
```

You should see a `luks` encrypted volume that uses the `/root/secret.key`.

4. Review block device details:

```
[root@HostD ~]# lsblk
```

The `luks` encrypted volume does not appear here.

5. Use `cryptsetup` to open the `luks-vgdata-lvdata` volume:

```
[root@HostD ~]# cryptsetup luksOpen /dev/mapper/vgdata-lvdata
luks-vgdata-lvdata --key-file /root/secret.key
```

You should see an error indicating there are no open keyslots. Therefore, `/root/secret.key` was removed from the available keyslots of the encrypted volume.

6. Restore the backup file. When prompted, type `YES`:

```
[root@HostD ~]# cryptsetup luksHeaderRestore /dev/mapper/
vgdata-lvdata --header-backup-file /opt/backups/vgdata-
lvdata.header
```

7. Re-run the `luksOpen` command:

```
[root@HostD ~]# cryptsetup luksOpen /dev/mapper/vgdata-lvdata
luks-vgdata-lvdata --key-file /root/secret.key
```

8. Reattempt to mount `/data`:

```
[root@HostD ~]# mount /data/
```

9. Review `/data/` to confirm the files are now there:

```
[root@HostD ~]# ls /data/
```

9. Troubleshoot `httpd`

1. On `HostB`, check the status of `httpd`:

```
[root@HostB ~]# systemctl status httpd
```

Observe it hasn't been started. Press **CTRL + C** to go back.

2. Start the service, and then check the status again:

```
[root@HostB ~]# systemctl start httpd
```

```
[root@HostB ~]# systemctl status httpd
```
This time there is a permissions issue related to your log file.

3. Review the `httpd` logs:

   ```
   [root@HostB ~]# ll /var/log/httpd
   ```
   There isn't too much helpful information returned.

4. Determine if SELinux is causing an issue:

   ```
   [root@HostB ~]# ausearch -m avc -ts recent
   ```
   Scroll up through the output, and at the top, observe a response regarding system context.

5. Check to see what the desired log type should be:

   ```
   [root@HostB ~]# semanage fcontext -l | grep /var/log/httpd
   ```

6. Check what the current context is:

   ```
   [root@HostB ~]# ls -Z /var/log | grep httpd
   ```
   It is currently using the `httpd_config` type rather than the `httpd_log` type.

7. Relabel the system's context:

   ```
   restorecon -Rv /var/log/httpd
   ```

8. Restart `httpd`, and check the status:

   ```
   [root@HostB ~]# systemctl start httpd
   [root@HostB ~]# systemctl status httpd
   ```
   This time there should be no errors.

Tasks 10–13

10. Enable Crash Dumps

On `HostA`, enable kernel crash dumps:

```
[root@HostA ~]# systemctl enable --now kdump
```

11. Resolve Communication between `HostD` and `HostA`

1. `HostD` cannot connect to the Apache server on `HostA` at `http://ip-10-0-1-101/index.html`. First, verify the issue from `HostD`:

   ```
   [root@HostD ~]# curl http://ip-10-0-1-101/index.html
   ```
   You should receive an error.

2. Attempt to use `curl` on only the IP address:

   ```
   [root@HostD ~]# curl 10.0.1.101
   ```
   It appears something is blocking the connection to this IP address.

3. Attempt to `ssh` in to the server. Use the password for `HostA`:

   ```
   [root@HostD ~]# ssh cloud_user@10.0.1.101
   ```

```
[root@HostD ~]# ssh cloud_user@10.0.1.101
```

4. Drop to the root user:

   ```
   [cloud_user@HostA ~]$ sudo -i
   ```

5. Check to see if port `80` is open:

   ```
   [root@HostA ~]# firewall-cmd --list-all
   ```

   It is not open.

6. Open the `httpd` port:

   ```
   [root@HostA ~]# firewall-cmd --permanent --zone=public --add-service=http
   ```

7. Reload the firewall:

   ```
   [root@HostA ~]# firewall-cmd --reload
   ```

8. Type `exit` twice to move back to the `HostD` root user.

9. Retry using `curl` to access only the IP address, and then try accessing the full address:

   ```
   [root@HostD ~]# curl 10.0.1.101
   ```

   This time it works!

   ```
   [root@HostD ~]# curl http://ip-10-0-1-101/index.html
   ```

   The error still appears for the full URL.

10. Review the `nsswitch` configuration file:

    ```
    [root@HostD ~]# cat /etc/nsswitch.conf
    ```

    Observe the `hosts` parameter does not currently reference `dns`, which is needed.

11. Check your `authselect` profiles:

    ```
    [root@HostD ~]# authselect current
    ```

    Currently, `sssd` is selected.

12. Ensure there are no other configurations under `/etc/sssd/conf.d`:

    ```
    [root@HostD ~]# ls /etc/sssd/conf.d
    ```

    You should see no configurations listed.

13. Edit the `nsswitch` configuration:

    ```
    [root@HostD ~]# vim /etc/nsswitch.conf
    ```

14. Go down to the `hosts` line, and add the `dns` option (press `i` to enter Insert mode). The edited line should look like this:

    ```
    hosts:      files dns myhostname
    ```

15. Press **ESC**, followed by `:wq` to save and quit.

16. Retry the page:

    ```
    [root@HostD ~]# curl http://ip-10-0-1-101/index.html
    ```

It should now work.

12. Resolve Samba Authentication Issues

1. On `HostC`, the `cloud_user` cannot list available Samba shares. First, replicate the issue. When prompted for a password, use `astrongpass`:

   ```
   [root@HostC ~]# smbclient -U cloud_user -L localhost
   ```

   You should see a failure.

2. Review some logs to try and find more information about the error:

   ```
   [root@HostC ~]# cat /var/log/samba/log.smbd
   ```

   Observe the PAM account validation failure.

3. Check your PAM configuration to see if a Samba file exists:

   ```
   [root@HostC ~]# ls /etc/pam.d/
   ```

   Observe a `samba` file in the directory.

4. Review the file:

   ```
   [root@HostC ~]# cat /etc/pam.d/samba
   ```

   Observe there is no configuration for logging in.

5. Verify the `samba` package:

   ```
   [root@HostC ~]# rpm -V samba
   ```

   From the results, note there have been some changes.

6. Move the file to create a backup:

   ```
   [root@HostC ~]# mv /etc/pam.d/samba /etc/pam.d/samba.bak
   ```

7. Reinstall the `samba` package:

   ```
   [root@HostC ~]# dnf reinstall samba -y
   ```

8. Review the configuration file again:

   ```
   [root@HostC ~]# cat /etc/pam.d/samba
   ```

   This time it contains two additional `auth` lines as well as an `account` line that was previously missing.

9. Reattempt to access Samba. When prompted for a password, use `astrongpass`:

   ```
   [root@HostC ~]# smbclient -U cloud_user -L localhost
   ```

   This time it is successful.

10. Remove the backup file. Press `y`, when prompted, to confirm deletion:

    ```
    [root@HostC ~]# rm /etc/pam.d/samba.bak
    ```


13. Discern Used Password

1. On `HostA`, observe that there is a `packet.pcap` file available:

```
[root@HostA ~]# cd /opt/reports/pcap/
[root@HostA ~]# ls
```

You'll need to open this file to determine the password used by `testuser01`.

2. Install `wireshark`:

```
[root@HostA ~]# dnf install wireshark -y
```

3. Use `tshark` to review `packet.pcap`:

```
[root@HostA ~]# tshark -r packet.pcap | less
```

Observe the contents relates to packet information — not user data. Press `q` to quit.

4. Review the `tshark` man page to see if there is a way you can get more information:

```
[root@HostA ~]# man tshark
```

Scroll through the man page, and observe the `-x` flag, which causes `tshark` to print a dump of packet data. This should provide more information about the packet.

5. Re-run `tshark` with this `-x` flag:

```
[root@HostA ~]# tshark -xr packet.pcap | less
```

This output contains a lot more information.

6. Search for `testuser` within the dump file:

```
/testuser
```

Observe there is a `password` field returned, and that the password is `mypaSSword`.

7. `echo` the password to the proper location:

```
[root@HostA ~]# echo "mypaSSword" > password
```

8. Verify the file was created:

```
[root@HostA ~]# cat password
```

Conclusion

Congratulations — you've completed the **Red Hat Certified Specialist in Linux Diagnostics and Troubleshooting (EX342)** practice exam. Best of luck!

Lab Tools

Lab Diagram | Instant Terminal

Lab Credentials

Help

It is recommended that all Hands-on Labs are opened in an **incognito window**.

Cloud Server - HostA

Username

cloud_user

Password

4q!26bUy

Private IP - HostA

10.0.1.101

Public IP - HostA

44.204.63.153

Cloud Server - HostB

Cloud Server - HostC

Cloud Server - HostD

Learning Objectives

Successfully complete this lab by achieving the following learning objectives.

1 Tasks 1–3

2 Tasks 4–6

3 Tasks 7–9

4 Tasks 10–13