Red Hat Enterprise Linux 7

Virtualization Security Guide

Securing hosts, guests, and shared infrastructure in virtualized environments on RHEL

Last Updated: 2019-09-02

Red Hat Enterprise Linux 7 Virtualization Security Guide

Securing hosts, guests, and shared infrastructure in virtualized environments on RHEL

Jiri Herrmann
Red Hat Customer Content Services
jherrman@redhat.com

Yehuda Zimmerman
Red Hat Customer Content Services
yzimmerm@redhat.com

Scott Radvan
Red Hat Customer Content Services

Tahlia Richardson
Red Hat Customer Content Services

Paul Moore
Red Hat Engineering

Kurt Seifried
Red Hat Engineering

David Jorm
Red Hat Engineering

Thanks go to the following people for enabling the creation of this guide:

Legal Notice

Abstract

This guide provides an overview of virtualization security technologies provided by Red Hat. It also provides recommendations for securing hosts, guests, and shared infrastructure and resources in virtualized environments.

Table of Contents

CHAPTER 1. INTRODUCTION

1.1. VIRTUALIZED AND NON-VIRTUALIZED ENVIRONMENTS

A virtualized environment presents opportunities for both the discovery of new attack vectors and the refinement of existing exploits that may not previously have presented value to an attacker. Therefore, it is important to take steps to ensure the security of both the physical hosts and the guests running on them when creating and maintaining virtual machines.

Non-Virtualized Environment

In a non-virtualized environment, hosts are separated from each other physically and each host has a self-contained environment, which consists of services such as a web server, or a DNS server. These services communicate directly to their own user space, host kernel and physical host, offering their services directly to the network.

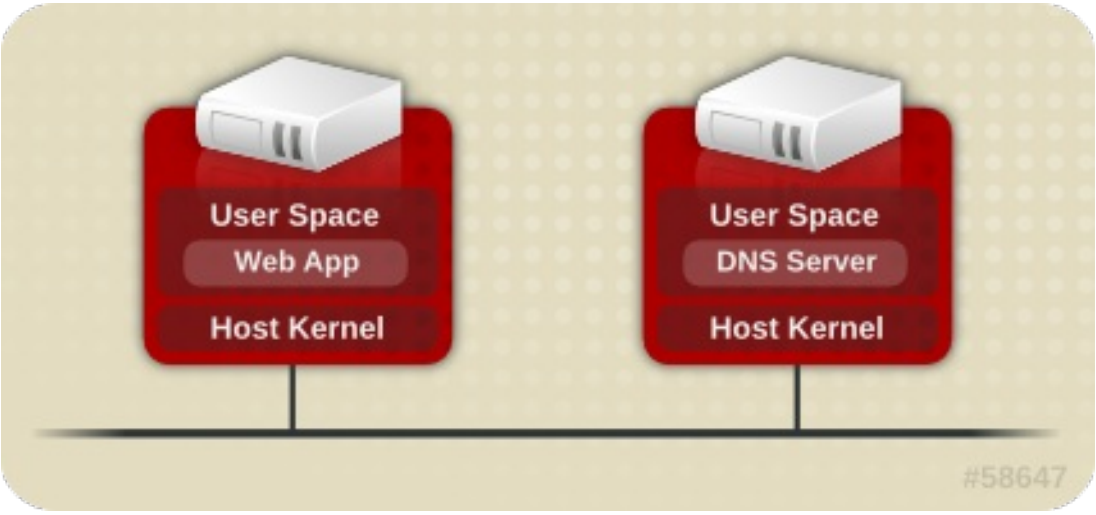


Figure 1.1. Non-Virtualized Environment

Virtualized Environment

In a virtualized environment, several operating systems can be housed (as guest virtual machines) within a single host kernel and physical host.

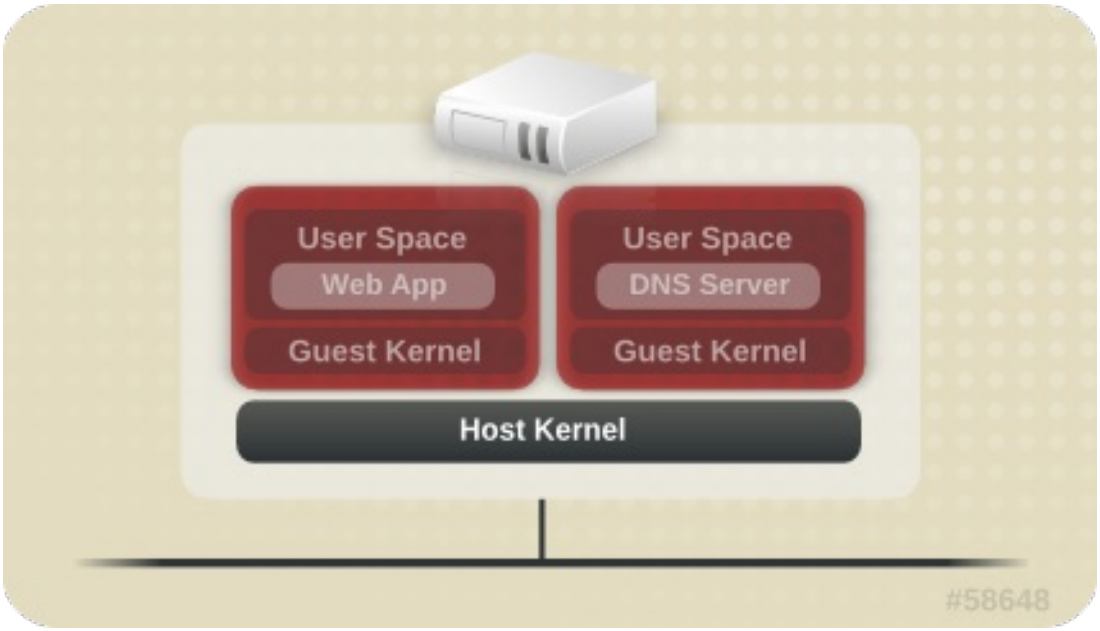


Figure 1.2. Virtualized Environment

When services are not virtualized, machines are physically separated. Any exploit is, therefore, usually contained to the affected machine, with the exception of network attacks. When services are grouped together in a virtualized environment, extra vulnerabilities emerge in the system. If a security flaw exists in the hypervisor that can be exploited by a guest instance, this guest may be able to attack the host, as well as other guests running on that host.

1.2. WHY VIRTUALIZATION SECURITY MATTERS

Deploying virtualization in your infrastructure provides many benefits, but can also introduce new risks. Virtualized resources and services should be deployed with the following security considerations:

- The host and hypervisor become prime targets; they are often a single point of failure for guests and data.
- Virtual machines can interfere with each other in undesirable ways. If no access controls are in place to help prevent this, one malicious guest can bypass a vulnerable hypervisor and directly access other resources on the host system, such as the storage of other guests.
- Resources and services can become difficult to track and maintain; with rapid deployment of virtualized systems comes an increased need for management of resources, including sufficient patching, monitoring and maintenance.
- Resources such as storage can be spread across, and dependent upon, several machines. This can lead to overly complex environments and poorly managed and maintained systems.
- Virtualization does not remove any of the traditional security risks present in your environment; the entire solution stack, not just the virtualization layer, must be secured.

This guide aims to assist you in mitigating your security risks by offering a number of virtualization recommended practices for Red Hat Enterprise Linux that will help you secure your virtualized infrastructure.

CHAPTER 2. HOST SECURITY

When deploying virtualization technologies on a Red Hat Enterprise Linux system, the host is responsible for managing and controlling access to the physical devices, storage, and network, but also to all virtualized guests. If the host system is compromised, the guests and their data become vulnerable as well.

Therefore, securing the Red Hat Enterprise Linux host system is the first step towards ensuring a secure virtualization platform.

2.1. SECURING THE HOST PHYSICAL MACHINE

The following tasks and tips can assist you with securing and ensuring reliability, as well increasing the performance, of your Red Hat Enterprise Linux host.

- Ensure that SELinux is configured properly for your installation and is operating in enforcing mode:

  ```
  # setenforce 1
  ```

  In addition to being a good security practice, the advanced virtualization security functionality provided by sVirt relies on SELinux. See Chapter 4, *sVirt* for more information on SELinux and sVirt.

- Remove or disable any unnecessary services such as **AutoFS**, **NFS**, **FTP**, **HTTP**, **NIS**, **telnetd**, or **sendmail**.

- Only add the minimum number of user accounts needed for platform management on the server and remove unnecessary user accounts. Limit direct access to the system to only those users who have a need to manage the system. Consider disallowing shared root access and instead use tools such as **sudo** to grant privileged access to administrators based on their administrative roles.

- Avoid running any unessential applications on your host. Running applications on the host may impact virtual machine performance and can affect server stability. Any application that may crash the server will also cause all virtual machines on the server to fail. In addition, vulnerable applications can become vectors for an attack on the host.

- Use a central location for virtual machine installations and images. Virtual machine images should be stored under **/var/lib/libvirt/images/**. If you are using a different directory for your virtual machine images make sure you add the directory to your SELinux policy and relabel it before starting the installation. Use of shareable, network storage in a central location is highly recommended.

- Run only the services necessary to support the use and management of your guest systems. If you need to provide additional services, such as file or print services, consider running those services on a Red Hat Enterprise Linux guest.

- Ensure that auditing is enabled on the host system and that libvirt is configured to generate audit records. When auditing is enabled, libvirt generates audit records for changes to guest configuration and start/stop events, which can help you track the guest's state. In addition, the libvirt audit events can also be viewed using the specialized **auvirt** utility. For more information, use the **man auvirt** command.

- Ensure that any remote management of the system takes place only over secured network

channels. Utilities such as SSH and network protocols such as TLS or SSL provide both authentication and data encryption to help ensure that only approved administrators can manage the system remotely.

- Ensure that the firewall is configured properly for your installation and is activated at boot. Only network ports needed for the use and management of the system should be allowed.
- Do not grant guests with direct access to entire disks or block devices (for example, **/dev/sdb**); instead, use partitions (for example, **/dev/sdb1**) or LVM volumes for guest storage.
- Attaching a USB device, Physical Function or physical device when SR-IOV is not available to a virtual machine could provide access to the device which is sufficient enough to overwrite that device's firmware. This presents a potential security issue by which an attacker could overwrite the device's firmware with malicious code and cause problems when moving the device between virtual machines or at host boot time.

  It is advised to use SR-IOV Virtual Function device assignment where applicable.

NOTE

For more security tips and instructions for your host system, see the *Red Hat Enterprise Linux Security Guide*.

2.2. CLIENT ACCESS CONTROL

libvirt's client access control framework allows system administrators to setup fine-grained permission rules across client users, managed objects, and API operations. This allows client connections to be locked down to a minimal set of privileges.

In the default configuration, the **libvirtd** daemon has three levels of access control:

1. All connections start off in an unauthenticated state, where the only API operations allowed are those required to complete authentication.
2. After successful authentication, a connection either has full, unrestricted access to all libvirt API calls, or is locked down to only "read only" operations, according to what socket the client connection originated on.
3. The access control framework allows authenticated connections to have fine-grained permission rules to be defined by the administrator.

Every API call in libvirt has a set of permissions that will be validated against the object being used. Further permissions will also be checked if certain flags are set in the API call. In addition to checks on the object passed in to an API call, some methods will filter their results.

2.2.1. Access Control Drivers

The access control framework is designed as a pluggable system to enable future integration with arbitrary access control technologies. By default, the **none** driver is used, which performs no access control checks at all. Currently, libvirt provides support for using polkit as a real access control driver. To learn how to use the polkit access driver see the configuration documentation.

The access driver is configured in the **/etc/libvirt/libvirtd.conf** configuration file, using the ***access_drivers*** parameter. This parameter accepts an array of access control driver names. If more than one access driver is requested, then all must succeed in order for access to be granted. To enable

'polkit' as the driver, use the **augtool** command:

```
# augtool -s set '/files/etc/libvirt/libvirtd.conf/access_drivers[1]' polkit
```

To set the driver back to the default (no access control), enter the following command:

```
# augtool -s rm /files/etc/libvirt/libvirtd.conf/access_drivers
```

For the changes made to **libvirtd.conf** to take effect, restart the **libvirtd** service.

```
# systemctl restart libvirtd.service
```

2.2.2. Objects and Permissions

libvirt applies access control to all the main object types in its API. Each object type, in turn, has a set of permissions defined. To determine what permissions are checked for a specific API call, consult the API reference manual documentation for the API in question. For the complete list of objects and permissions, see libvirt.org.

2.2.3. Security Concerns when Adding Block Devices to a Guest

- The host physical machine should not use file system labels to identify file systems in the **fstab** file, the **initrd** file or on the kernel command line. Doing so presents a security risk if guest virtual machines have write access to whole partitions or LVM volumes, because a guest virtual machine could potentially write a file-system label belonging to the host physical machine, to its own block device storage. Upon reboot of the host physical machine, the host physical machine could then mistakenly use the guest virtual machine's disk as a system disk, which would compromise the host physical machine system.

  It is preferable to use the UUID of a device to identify it in the **/etc/fstab** file, the **/dev/initrd** file, or on the kernel command line.

- Guest virtual machines should not be given write access to entire disks or block devices (for example, **/dev/sdb**). Guest virtual machines with access to entire block devices may be able to modify volume labels, which can be used to compromise the host physical machine system. Use partitions (for example, **/dev/sdb1**) or LVM volumes to prevent this problem. See LVM Administration with CLI Commands or LVM Configuration Examples for information on LVM administration and configuration examples.

  If you are using raw access to partitions, for example **/dev/sdb1** or raw disks such as **/dev/sdb,** you should configure LVM to only scan disks that are safe, using the **global_filter** setting. See the Logical Volume Manager Administration Guide for an example of an LVM configuration script using the **global_filter** command.

2.3. SPECIAL CONSIDERATIONS FOR PUBLIC CLOUD OPERATORS

Public cloud service providers are exposed to a number of security risks beyond that of the traditional virtualization user. Virtual guest isolation, both between the host and guest as well as between guests, is critical due to the threat of malicious guests and the requirements on customer data confidentiality and integrity across the virtualization infrastructure.

In addition to the Red Hat Enterprise Linux virtualization recommended practices previously listed, public cloud operators should also consider the following items:

- Disallow any direct hardware access from the guest. PCI, USB, FireWire, Thunderbolt, eSATA, and other device passthrough mechanisms make management difficult and often rely on the underlying hardware to enforce separation between the guests.
- Isolate the cloud operator's private management network from the customer guest network, and customer networks from one another, so that:
  - The guests cannot access the host systems over the network.
  - One customer cannot access another customer's guest systems directly through the cloud provider's internal network.

CHAPTER 3. GUEST SECURITY

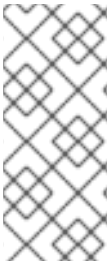
3.1. WHY GUEST SECURITY MATTERS

While the security of the host system is critical in ensuring the security of the guests running on the host, it does not remove the need for properly securing the individual guest machines. All of the security risks associated with a conventional, non-virtualized system still exist when the system is run as a virtualized guest. Any resources accessible to the guest system, such as critical business data or sensitive customer information, could be made vulnerable if the guest system were to be compromised.

3.2. GUEST SECURITY RECOMMENDED PRACTICES

All of the recommended practices for securing a Red Hat Enterprise Linux system documented in the *Red Hat Enterprise Linux Security Guide* apply to conventional, non-virtualized systems as well as systems installed as a virtualized guest. However, there are a few security practices which are of critical importance when running guests in a virtualized environment:

- With all management of the guest likely taking place remotely, ensure that the management of the system takes place only over secured network channels. Tools such as SSH and network protocols such as TLS or SSL provide both authentication and data encryption to ensure that only approved administrators can manage the system remotely.
- Some virtualization technologies use special guest agents or drivers to enable some virtualization specific features. Ensure that these agents and applications are secured using the standard Red Hat Enterprise Linux security features, such as SELinux.
- In virtualized environments, a greater risk exists of sensitive data being accessed outside the protection boundaries of the guest system. Protect stored sensitive data using encryption tools such as **dm-crypt** and **GnuPG**; although special care needs to be taken to ensure the confidentiality of the encryption keys.



NOTE

Using page deduplication technology such as Kernel Same-page Merging (KSM) may introduce side channels that could potentially be used to leak information across guests. In situations where this is a concern, KSM can be disabled either globally or on a per-guest basis. For more information about KSM, see the Red Hat Enterprise Linux 7 Virtualization Tuning and Optimization Guide.

3.3. KERNEL ADDRESS SPACE RANDOMIZATION

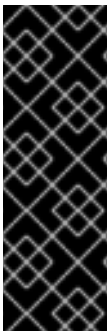
Red Hat Enterprise Linux 7.5 and later include the Kernel Address Space Randomization (KASLR) feature for KVM guest virtual machines. KASLR enables randomizing the physical and virtual address at which the kernel image is decompressed, and thus prevents guest security exploits based on the location of kernel objects.

KASLR is activated by default, but can be deactivated on a specific guest by adding the **nokaslr** string to the guest's kernel command line. To edit the guest boot options, use the following command, where *guestname* is the name of your guest:

```
# virt-edit -d guestname /etc/default/grub
```

Afterwards, modify the **GRUB_CMDLINE_LINUX** line, for example:

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet nokaslr"
```



IMPORTANT

Guest dump files created from guests that have with KASLR activated are not readable by the **crash** utility. To fix this, add the **<vmcoreinfo/>** element to the **<features>** section of the XML configuration files of your guests.

Note, however, that migrating guests with **<vmcoreinfo/>** fails if the destination host is using an OS that does not support **<vmcoreinfo/>**. These include Red Hat Enterprise Linux 7.4 and earlier, as well as Red Hat Enterprise Linux 6.9 and earlier.

CHAPTER 4. SVIRT

4.1. INTRODUCTION

Since virtual machines under KVM are implemented as Linux processes, KVM uses the standard Linux security model to provide isolation and resource controls. The Linux kernel includes Security-Enhanced Linux (SELinux) to add mandatory access control (MAC), multi-level security (MLS) and multi-category security (MCS) through a flexible and customizable security policy. SELinux provides strict resource isolation and confinement for processes running on top of the Linux kernel, including virtual machine processes. The sVirt project builds upon SELinux to further enable virtual machine isolation and controlled sharing. For example, fine-grained permissions could be applied to group virtual machines together to share resources.

From a security point of view, the hypervisor is a likely target for attackers, as a compromised hypervisor can lead to the all virtual machines running on the host system. Integrating SELinux into virtualization technologies helps improve hypervisor security against malicious virtual machines trying to gain access to the host system or other virtual machines.

The following image represents SELinux isolating guests, which limits the ability for a compromised hypervisor (or guest) to launch further attacks, or to extend to another instance:

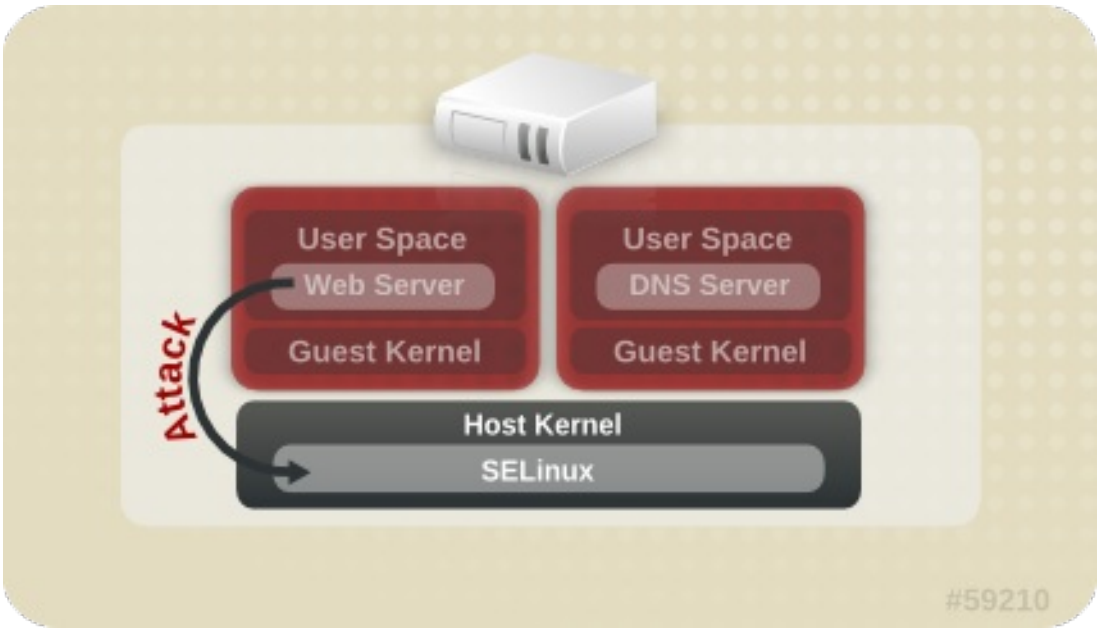


Figure 4.1. Attack path isolated by SELinux

NOTE

For more information on SELinux, refer to the Red Hat Enterprise Linux SELinux Users and Administrators Guide.

4.2. SELINUX AND MANDATORY ACCESS CONTROL (MAC)

Security-Enhanced Linux (SELinux) is an implementation of MAC in the Linux kernel, checking for allowed operations after standard discretionary access controls (DAC) are checked. SELinux can enforce a user-customizable security policy on running processes and their actions, including attempts to access file system objects. Enabled by default in Red Hat Enterprise Linux, SELinux limits the scope of potential damage that can result from the exploitation of vulnerabilities in applications and system services, such as the hypervisor.

sVirt integrates with libvirt, a virtualization management abstraction layer, to provide a MAC framework for virtual machines. This architecture allows all virtualization platforms supported by libvirt and all MAC implementations supported by sVirt to interoperate.

4.3. SVIRT CONFIGURATION

SELinux Booleans are variables that can be toggled on or off, quickly enabling or disabling features or other special conditions. Booleans can be toggled by running either **setsebool *boolean_name* {on|off}** for a temporary change, or **setsebool -P *boolean_name* {on|off}** to make the change persistent across reboots.

The following table shows the SELinux Boolean values that affect KVM when launched by libvirt. The current state of these booleans (on or off) can be found by running the command **getsebool -a|grep virt**.

Table 4.1. KVM SELinux Booleans

SELinux Boolean	Description
staff_use_svirt	Enables staff users to create and transition to sVirt domains.
unprivuser_use_svirt	Enables unprivileged users to create and transition to sVirt domains.
virt_sandbox_use_audit	Enables sandbox containers to send audit messages.
virt_sandbox_use_netlink	Enables sandbox containers to use netlink system calls.
virt_sandbox_use_sys_admin	Enables sandbox containers to use sys_admin system calls, such as mount.
virt_transition_userdomain	Enables virtual processes to run as user domains.
virt_use_comm	Enables virt to use serial/parallel communication ports.
virt_use_execmem	Enables confined virtual guests to use executable memory and executable stack.
virt_use_fusefs	Enables virt to read FUSE mounted files.
virt_use_nfs	Enables virt to manage NFS mounted files.
virt_use_rawip	Enables virt to interact with rawip sockets.
virt_use_samba	Enables virt to manage CIFS mounted files.

SELinux Boolean	Description
virt_use_sanlock	Enables confined virtual guests to interact with the sanlock.
virt_use_usb	Enables virt to use USB devices.
virt_use_xserver	Enables virtual machine to interact with the X Window System.

NOTE

For more information on SELinux Booleans, see the Red Hat Enterprise Linux SELinux Users and Administrators Guide.

4.4. SVIRT LABELING

Like other services under the protection of SELinux, sVirt uses process-based mechanisms, labels, and restrictions to provide extra security and control over guest instances. Labels are applied automatically to resources on the system based on the currently running virtual machines (dynamic), but can also be manually specified by the administrator (static), to meet any specific requirements that may exist.

To edit the sVirt label of a guest, use the **virsh edit *guest_name*** command and add or edit **<seclabel>** elements as described in the sections below. **<seclabel>** can be used as a root element for the entire guest, or it can be specified as a sub-element of the **<source>** element for selecting a specific sVirt label of the given device.

For comprehensive information about the **<seclabel>** element, see the libvirt upstream documentation.

4.4.1. Types of sVirt Labels

The following table outlines the different sVirt labels that can be assigned to resources such as virtual machine processes, image files and shared content:

Table 4.2. sVirt Labels

Type	SELinux Context	Description/Effect
Virtual Machine Processes	system_u:system_r:svirt_t:*MCS1*	*MCS1* is a randomly selected field. Currently approximately 500,000 labels are supported.
Virtual Machine Image	system_u:object_r:svirt_image_t:*MCS1*	Only *svirt_t* processes with the same *MCS1* fields are able to read/write these image files and devices.
Virtual Machine Shared Read/Write Content	system_u:object_r:svirt_image_t:s0	All *svirt_t* processes are allowed to write to the *svirt_image_t:s0* files and devices.

Type	SELinux Context	Description/Effect
Virtual Machine Shared Shared Read Only content	system_u:object_r:svirt_content_t:s0	All *svirt_t* processes are able to read files/devices with this label.
Virtual Machine Image	system_u:object_r:virt_content_t:s0	System default label used when an image exits. No *svirt_t* virtual processes are allowed to read files/devices with this label.

4.4.2. Dynamic Configuration

Dynamic label configuration is the default labeling option when using sVirt with SELinux. See the following example which demonstrates dynamic labeling:

```
# ps -eZ | grep qemu-kvm

system_u:system_r:svirt_t:s0:c87,c520 27950 ?  00:00:17 qemu-kvm
```

In this example, the **qemu-kvm** process has a base label of **system_u:system_r:svirt_t:s0**. The libvirt system has generated a unique MCS label of **c87,c520** for this process. The base label and the MCS label are combined to form the complete security label for the process. Likewise, libvirt takes the same MCS label and base label to form the image label. This image label is then automatically applied to all host files that the VM is required to access, such as disk images, disk devices, PCI devices, USB devices, and kernel/initrd files. Each process is isolated from other virtual machines with different labels.

The following example shows the virtual machine's unique security label (with a corresponding MCS label of **c87,c520** in this case) as applied to the guest disk image file in **/var/lib/libvirt/images**:

```
# ls -lZ /var/lib/libvirt/images/*

  system_u:object_r:svirt_image_t:s0:c87,c520   image1
```

The following example shows dynamic labeling in the XML configuration for the guest:

```
<seclabel type='dynamic' model='selinux' relabel='yes'>
  <label>system_u:system_r:svirt_t:s0:c87,c520</label>
  <imagelabel>system_u:object_r:svirt_image_t:s0:c87,c520</imagelabel>
</seclabel>
```

4.4.3. Dynamic Configuration with Base Labeling

To override the default base security label in dynamic mode, the **<baselabel>** option can be configured manually in the XML guest configuration, as shown in this example:

```
<seclabel type='dynamic' model='selinux' relabel='yes'>
  <baselabel>system_u:system_r:svirt_custom_t:s0</baselabel>
  <label>system_u:system_r:svirt_custom_t:s0:c87,c520</label>
  <imagelabel>system_u:object_r:svirt_image_t:s0:c87,c520</imagelabel>
</seclabel>
```

4.4.4. Static Configuration with Dynamic Resource Labeling

Some applications require full control over the generation of security labels but still require libvirt to take care of resource labeling. The following guest XML configuration demonstrates an example of static configuration with dynamic resource labeling:

```
<seclabel type='static' model='selinux' relabel='yes'>
  <label>system_u:system_r:svirt_custom_t:s0:c87,c520</label>
</seclabel>
```

4.4.5. Static Configuration without Resource Labeling

Primarily used in multi-level security (MLS) and other strictly controlled environments, static configuration without resource relabeling is possible. Static labels allow the administrator to select a specific label, including the MCS/MLS field, for a virtual machine. Administrators who run statically-labeled virtual machines are responsible for setting the correct label on the image files. The virtual machine will always be started with that label, and the sVirt system will never modify the label of a statically-labelled virtual machine's content. The following guest XML configuration demonstrates an example of this scenario:

```
<seclabel type='static' model='selinux' relabel='no'>
  <label>system_u:system_r:svirt_custom_t:s0:c87,c520</label>
</seclabel>
```

4.4.6. sVirt Labeling and NFS

To use sVirt labeling on a NFSv4.1 or NFSv4.2 file system, you need to change the SELinux context to **virt_var_lib_t** for the root of the NFS directory that you are exporting for guest sharing. For example, if you are exporting the **/exports/nfs/** directory, use the following commands:

```
# semanage fcontext -a -t virt_var_lib_t '/exports/nfs/'
# restorecon -Rv /exports/nfs/
```

In addition, when **libvirt** dynamically generates an sVirt label for a guest virtual machines on a NFS volume, it only guarantees label uniqueness within a single host. This means that if a high number of guests across multiple hosts share a NFS volume, it is possible for duplicate labels to occur, which creates a potential vulnerability.

To avoid this situation, do one of the following:

- Use a different NFS volume for each virtualization host. In addition, when performing guest migration, copy the guest storage by using the **--migrate-disks** and **--copy-storage-all** options.
- When creating a new guest with the **virt-install** command, set a static label for the guest by:
  - Using the **--security** option. For example:

    ```
    # virt-install --name guest1-rhel7 --memory 2048 --vcpus 2 --disk size=8 --cdrom
    /home/username/Downloads/rhel-workstation-7.4-x86_64-dvd.iso --os-variant rhel7 --
    security model=selinux,label='system_u:object_r:svirt_image_t:s0:c100,c200'
    ```

    This sets the security label for all disks on the guest.

  - Using the **--disk** option with the ***seclabel*** parameter. For example:

```
# virt-install --name guest1-rhel7 --memory 2048 --vcpus 2 --disk
/path/to/disk.img,seclabel.model=selinux,seclabel.label='system_u:object_r:svirt_image_t:s0
c100,c200' --cdrom /home/username/Downloads/rhel-workstation-7.4-x86_64-dvd.iso --
os-variant rhel7
```

This sets the security label only on the specified disks.

CHAPTER 5. NETWORK SECURITY IN A VIRTUALIZED ENVIRONMENT

5.1. NETWORK SECURITY OVERVIEW

In almost all situations, the network is the only way to access systems, applications, and management interfaces. As networking plays such a critical role in the management of virtualized systems and the availability of their hosted applications, it is very important to ensure that the network channels both to and from the virtualized systems are secure.

Securing the network allows administrators to control access and protect sensitive data from information leaks and tampering.

5.2. NETWORK SECURITY RECOMMENDED PRACTICES

Network security is a critical part of a secure virtualization infrastructure. See the following recommended practices for securing the network:

- Ensure that remote management of the system takes place only over secured network channels. Tools such as SSH and network protocols such as TLS or SSL provide both authentication and data encryption to assist with secure and controlled access to systems.
- Ensure that guest applications transferring sensitive data do so over secured network channels. If protocols such as TLS or SSL are not available, consider using one like IPsec.
- Configure firewalls and ensure they are activated at boot. Only network ports needed for the use and management of the system should be allowed. Test and review firewall rules regularly.

5.2.1. Securing Connectivity to SPICE

The SPICE remote desktop protocol supports SSL/TLS which should be enabled for all of the SPICE communication channels (main, display, inputs, cursor, playback, record).

5.2.2. Securing Connectivity to Storage

You can connect virtualized systems to networked storage in many different ways. Each approach presents different security benefits and concerns, but the same security principles apply to each: authenticate the remote store pool before use, and protect the confidentiality and integrity of the data while it is being transferred.

The data must also remain secure while it is stored. Red Hat recommends that data is encrypted or digitally signed before storing, or both.

NOTE

For more information on networked storage, see the Storage Pools chapter of the *Red Hat Enterprise Linux Virtualization Deployment and Administration Guide*.

APPENDIX A. FURTHER INFORMATION

A.1. SELINUX AND SVIRT

Further information on SELinux and sVirt:

- Main SELinux website: https://www.nsa.gov/what-we-do/research/selinux/documentation/assets/files/presentations/2004-ottawa-linux-symposium-bof-presentation.pdf.
- SELinux documentation: https://www.nsa.gov/what-we-do/research/selinux/documentation/index.shtml.
- Main sVirt website: http://selinuxproject.org/page/SVirt.
- Dan Walsh's blog: http://danwalsh.livejournal.com/.

A.2. VIRTUALIZATION SECURITY

Further information on virtualization security:

- NIST (National Institute of Standards and Technology) full virtualization security guidelines: http://www.nist.gov/itl/csd/virtual-020111.cfm.

APPENDIX B. REVISION HISTORY

Revision 1.0-22 Version for 7.7 Beta publication	**Thu May 23 2019**	**Jiri Herrmann**
Revision 1.0-21 Version for 7.6 GA publication	**Thu Oct 25 2018**	**Jiri Herrmann**
Revision 1.0-21 Version for 7.6 Beta publication	**Thu Aug 14 2018**	**Jiri Herrmann**
Revision 1.0-20 Version for 7.5 GA publication	**Thu Apr 5 2018**	**Jiri Herrmann**
Revision 1.0-18 Version for 7.4 GA publication	**Thu Jul 27 2017**	**Jiri Herrmann**
Revision 1.0-15 Version for 7.3 GA publication	**Mon Oct 17 2016**	**Jiri Herrmann**
Revision 1.0-9 Cleaned up the Revision History	**Thu Oct 08 2015**	**Jiri Herrmann**
Revision 1.0-8 Version for 7.1 GA release.	**Wed Feb 18 2015**	**Scott Radvan**