

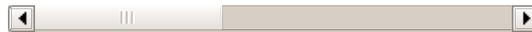


Fedora 19

Virtualization Security Guide

Virtualization Documentation

Edition 0.2



Red Hat Engineering Content Services

Legal Notice

Copyright © 2013 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at <http://creativecommons.org/licenses/by-sa/3.0/>. The original authors of this document, and Red Hat, designate the Fedora Project as the "Attribution Party" for purposes of CC-BY-SA. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

For guidelines on the permitted uses of the Fedora trademarks, refer to https://fedoraproject.org/wiki/Legal:Trademark_guidelines.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

All other trademarks are the property of their respective owners.

Abstract

This guide provides an overview of virtualization security technologies

provided by Fedora, and provides recommendations for securing hosts, guests, and shared infrastructure and resources in virtualized environments.

Preface

1. Document Conventions

- 1.1. Typographic Conventions
- 1.2. Pull-quote Conventions
- 1.3. Notes and Warnings

2. We Need Feedback!

1. Introduction

- 1.1. Virtualized and Non-Virtualized Environments
- 1.2. Why Virtualization Security Matters
- 1.3. Three Way Model
- 1.4. Leveraging SELinux with sVirt

2. Host Security

- 2.1. Why Host Security Matters
- 2.2. Host Security Best Practices for Fedora
 - 2.2.1. Special Considerations for Public Cloud Operators

3. Guest Security

- 3.1. Why Guest Security Matters
- 3.2. Guest Security Best Practices

4. sVirt

- 4.1. Introduction
- 4.2. SELinux and Mandatory Access Control (MAC)
- 4.3. sVirt Configuration
- 4.4. sVirt Labeling
 - 4.4.1. Types of sVirt Labels
 - 4.4.2. Dynamic Configuration
 - 4.4.3. Dynamic Configuration with Base Labeling
 - 4.4.4. Static Configuration with Dynamic Resource Labeling
 - 4.4.5. Static Configuration without Resource Labeling

5. Network Security in a Virtualized Environment

- 5.1. Network Security Overview
- 5.2. Network Security Best Practices
 - 5.2.1. Securing Connectivity to Spice
 - 5.2.2. Securing Connectivity to Storage

6. Further Information

- 6.1. Contributors
- 6.2. Other Resources

A. Revision History

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the [Liberation Fonts](#) set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

Mono-spaced Bold

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

To see the contents of the file **my_next_bestselling_novel** in your current working directory, enter the **cat my_next_bestselling_novel** command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

Press **Enter** to execute the command.

Press **Ctrl+Alt+F2** to switch to the first virtual terminal.
Press **Ctrl+Alt+F1** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **mono-spaced bold**. For example:

File-related classes include **filesystem** for file systems, **file** for files, and **dir** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find...** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the

Text to copy field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username@domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books      Desktop  documentation  drafts  mss
photos    stuff  svn
books_tests Desktop1  downloads      images  notes
scripts   svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
    public static void main(String args[])
        throws Exception
    {
        InitialContext iniCtx = new InitialContext();
        Object ref =
iniCtx.lookup("EchoBean");
        EchoHome home = (EchoHome) ref;
        Echo echo = home.create();

        System.out.println("Created Echo");

        System.out.println("Echo.echo('Hello') = " +
echo.echo("Hello"));
    }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.



Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.



Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: <http://bugzilla.redhat.com/bugzilla/> against the product **Fedora 19**.

When submitting a bug report, be sure to mention the manual's identifier: *virtualization-security-guide*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Chapter 1. Introduction

1.1. Virtualized and Non-Virtualized Environments

1.2. Why Virtualization Security Matters

1.3. Three Way Model

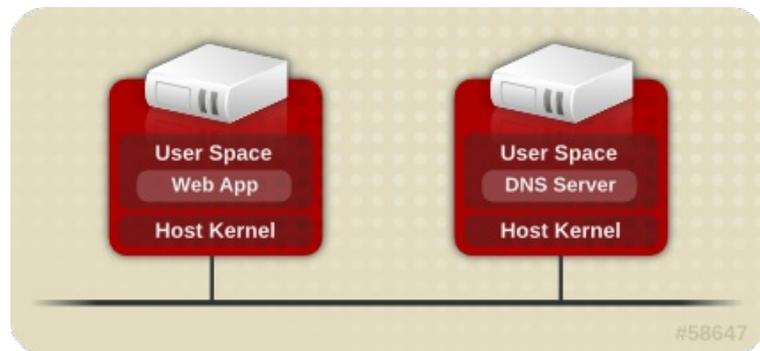
1.4. Leveraging SELinux with sVirt

1.1. Virtualized and Non-Virtualized Environments

A virtualized environment presents the opportunity for both the discovery of new attack vectors and the refinement of existing exploits which may not previously have presented value to an attacker. It is therefore 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.

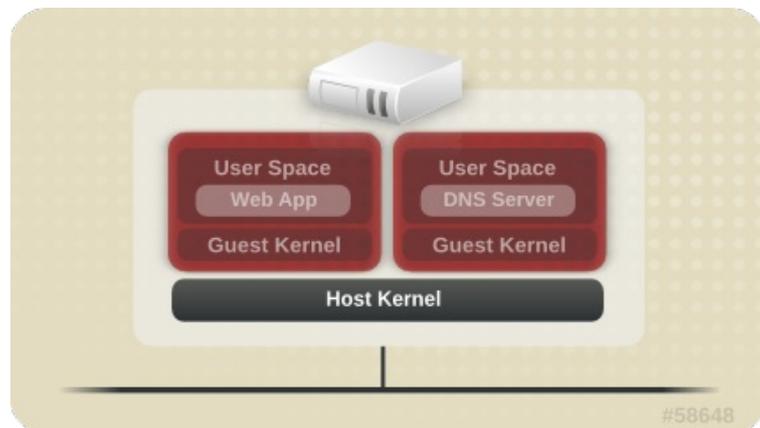
Non-Virtualized Environment

In a non-virtualized environment, hosts are separated from each other physically and each host has a self-contained environment, consisting 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. The following image represents a non-virtualized environment:



Virtualized Environment

In a virtualized environment, several operating systems can be housed (as "guests") within a single host kernel and physical host. The following image represents a virtualized environment:



When services are not virtualized, machines are physically separated. Any exploit is therefore usually contained to the affected machine, with the obvious exception of network attacks. When services are grouped

together in a virtualized environment, extra vulnerabilities emerge in the system. If there is a security flaw in the hypervisor that can be exploited by a guest instance, this guest may be able to not only attack the host, but also other guests running on that host. This is not theoretical; attacks already exist on hypervisors. These attacks can extend beyond the guest instance and could expose other guests to attack.

1.2. Why Virtualization Security Matters

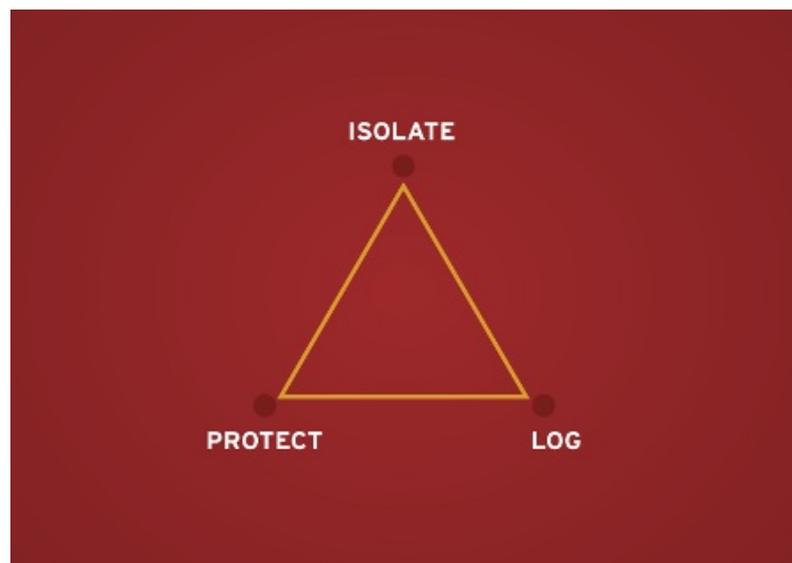
Deploying virtualization in your infrastructure provides many benefits but can also introduce new risks. Virtualized resources and services should be deployed with the following security considerations:

- ▶ The host/hypervisor become prime targets; in effect, they are often a single point of failure for guests and data.
- ▶ Virtual machines can interfere with each other in undesirable ways.
- ▶ 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.
- ▶ There may be a lack of knowledge, gaps in skill sets, and minimal experience among technical staff. This is often a gateway to vulnerabilities.
- ▶ 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 best practices for Fedora that will help you secure your virtualized infrastructure.

1.3. Three Way Model

A three-way conceptual model, CIA (Confidentiality, Integrity, Availability) is often used in general computing security. A similar model can be presented in addition to this model when analyzing virtualization security: IPL (Isolate, Protect, Log), as shown in the following diagram:



Isolate

Controlling interactions between virtual machines is crucial to maintaining a high level of security. This is provided in Fedora by sVirt.

Protect

Virtualized machines are not immune to traditional security threats. Each virtual machine should be managed with regular security controls.

Log

Virtual machines are simple to deploy. The lack of logging, change management and audit trails in a virtualized environment can easily lead to a sprawling, unmanaged and insecure environment.

1.4. Leveraging SELinux with sVirt

sVirt integrates virtualization into the existing security framework provided by SELinux (Security-Enhanced Linux), applying *Mandatory Access Control* (MAC) to virtual machines. The main objective of sVirt is to protect hosts and guests from attacks via security vulnerabilities in the hypervisor. SELinux secures a system by applying access policy across different processes. sVirt extends this capability to hosts and guests by treating each guest as a process, allowing administrators to apply similar policies designed to prevent malicious guests from accessing restricted resources. For more information on sVirt, refer to [Chapter 4, sVirt](#).

Chapter 2. Host Security

2.1. Why Host Security Matters

2.2. Host Security Best Practices for Fedora

2.2.1. Special Considerations for Public Cloud Operators

2.1. Why Host Security Matters

When deploying virtualization technologies, the security of the host should be paramount. The Fedora host system is responsible for managing and controlling access to the physical devices, storage and network as well as all virtualized guests themselves. If the host system were to be compromised, not only would the host system be vulnerable, but the guests and their data would be also.

Virtualized guests are only as secure as their host system; securing the Fedora host system is the first step towards ensuring a secure virtualization platform.

2.2. Host Security Best Practices for Fedora

With host security being such a critical part of a secure virtualization infrastructure, the following best practices should serve as a starting point for securing a Fedora host system:

- ▶ 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, you should consider running those services in a Fedora guest.
- ▶ 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.
- ▶ Ensure that SELinux is configured properly for your installation and is operating in enforcing mode. Besides being a good security practice, the advanced virtualization security functionality provided by sVirt relies on SELinux. Refer to [Chapter 4, sVirt](#) for more information on SELinux and sVirt.
- ▶ Ensure that any 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 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 those network ports needed for the use and management of the system should be allowed.
- ▶ Refrain from granting guests direct access to entire disks or block devices (for example, `/dev/sdb`); instead, you should use partitions (for example, `/dev/sdb1`) or LVM volumes for guest storage.

Note

The objective of this guide is to explain the unique security related challenges, vulnerabilities, and solutions that are present in most virtualized environments and how to best address them. However, there are a number of best practices to follow when securing a Fedora system that apply regardless of whether the system is a standalone, virtualization host, or guest instance. These best practices include procedures such as system updates, password security, encryption, and firewall configuration. This information is discussed in more detail in the *Fedora Security Guide* which can be found at <http://docs.fedoraproject.org>.

2.2.1. 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, as well as the requirements on customer data confidentiality and integrity across the virtualization infrastructure.

In addition to the Fedora virtualization best 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 not only make management difficult, but often rely on the underlying hardware to enforce separation between the guests.
- ▶ Network traffic should be separated such that the cloud operator's private management network is isolated from the customer guest network, helping to ensure that the guests can not access the host systems over the network. The network should be further isolated such that customer guest systems run in a private, virtualized network so that one customer can not access another customer's guest systems directly via the cloud provider's internal network.

Chapter 3. Guest Security

3.1. Why Guest Security Matters

3.2. Guest Security Best Practices

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 Best Practices

All of the best practices for securing a Fedora system documented in the *Fedora Security Guide* apply to the conventional, non-virtualized systems as well as those systems installed as a virtualized guest. However, there are a few security practices which are of critical importance when running 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 Fedora security features, e.g. SELinux.
- ▶ In virtualized environments there is a greater risk 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.

Chapter 4. sVirt

4.1. Introduction

4.2. SELinux and Mandatory Access Control (MAC)

4.3. sVirt Configuration

4.4. sVirt Labeling

4.4.1. Types of sVirt Labels

4.4.2. Dynamic Configuration

4.4.3. Dynamic Configuration with Base Labeling

4.4.4. Static Configuration with Dynamic Resource Labeling

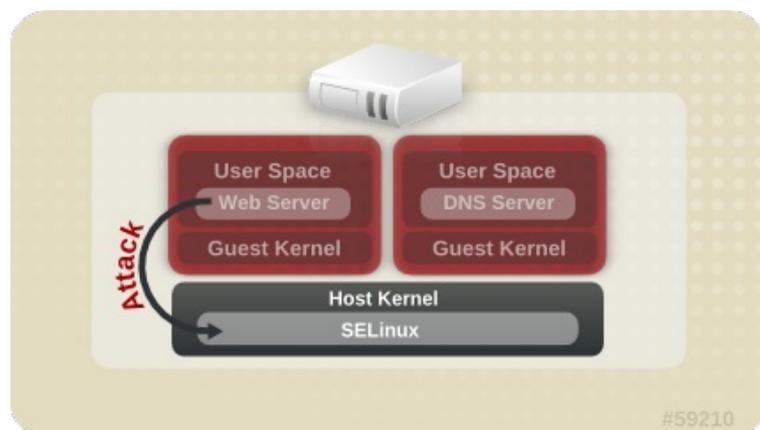
4.4.5. Static Configuration without Resource Labeling

4.1. Introduction

Since virtual machines under KVM are implemented as Linux processes, KVM leverages the standard Linux security model to provide isolation and resource controls. The Linux kernel includes SELinux (Security-Enhanced Linux), a project developed by the US National Security Agency 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 facilitate 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 tempting target for attackers, as a compromised hypervisor could lead to the compromise of all virtual machines running on the host system. Integrating SELinux into the virtualization technologies helps improve the security of the hypervisor against malicious virtual machines trying to gain access to the host system or other virtual machines.

Refer to the following image which represents isolated guests, limiting the ability for a compromised hypervisor (or guest) to launch further attacks, or to extend to another instance:



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 their attempts to access file system objects. Enabled by default in Fedora, 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. The following table shows the SELinux Boolean values that affect KVM when launched by libvirt.

KVM SELinux Booleans

SELinux Boolean	Description
virt_use_comm	Allow virt to use serial/parallel communication ports.
virt_use_fusefs	Allow virt to read FUSE mounted files.
virt_use_nfs	Allow virt to manage NFS mounted files.
virt_use_samba	Allow virt to manage CIFS mounted files.
virt_use_sanlock	Allow sanlock to manage virt lib files.
virt_use_sysfs	Allow virt to manage device configuration (PCI).
virt_use_usb	Allow virt to use USB devices.
virt_use_xserver	Allow virtual machine to interact with the X Window System.

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.

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 4.1. 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 <i>svirt_t</i> 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 <i>svirt_t</i> processes are allowed to write to the <i>svirt_image_t:s0</i> files and devices.
Virtual Machine Shared Shared Read Only content	system_u:object_r:svirt_content_t:s0	All <i>svirt_t</i> 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 exists. No <i>svirt_t</i> 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. Refer to 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</baselab
el>

<label>system_u:system_r:svirt_custom_t:s0:c87,c520</la
bel>

<imagelabel>system_u:object_r:svirt_image_t:s0:c87,c52
0</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</la
bel>
</seclabel>
```

4.4.5. Static Configuration without Resource Labeling

Primarily used in MLS (multi-level security) or otherwise 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</la
bel>
</seclabel>
```

Chapter 5. Network Security in a Virtualized Environment

5.1. Network Security Overview

5.2. Network Security Best Practices

5.2.1. Securing Connectivity to Spice

5.2.2. Securing Connectivity to Storage

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 Best Practices

Network security is a critical part of a secure virtualization infrastructure. Refer to the following best 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 IPsec.
- ▶ Configure firewalls and ensure they are activated at boot. Only those 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, however 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. It is considered a best practice to encrypt and/or digitally sign the data before storing it.

Chapter 6. Further Information

6.1. Contributors

6.2. Other Resources

6.1. Contributors

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

- ▶ Paul Moore - Red Hat Engineering
- ▶ Kurt Seifried - Red Hat Engineering
- ▶ David Jorm - Red Hat Engineering

6.2. Other Resources

SELinux and sVirt

Further information on SELinux and sVirt:

- ▶ Main SELinux website: <http://www.nsa.gov/research/selinux/index.shtml>.
- ▶ SELinux documentation: <http://www.nsa.gov/research/selinux/docs.shtml>.
- ▶ Main sVirt website: <http://selinuxproject.org/page/SVirt>.
- ▶ Dan Walsh's blog: <http://danwalsh.livejournal.com/>.
- ▶ The unofficial SELinux FAQ: <http://www.crypt.gen.nz/selinux/faq.html>.

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>.

Revision History

Revision 0.2-01	Thu Jun 13 2013	Scott Radvan
Fedora 19 GA publish.		
Revision 0.1-04	Mon Aug 27 2012	Scott Radvan
Prepare for draft publish.		
Revision 0.1-03	Thu Aug 16 2012	Scott Radvan
Expand IPL model. Fix .ent file to show correct BZ component for Fedora.		
Revision 0.1-02	Thu Aug 16 2012	Scott Radvan
Fork from Red Hat guide.		

