IBM

Blueprints

Installing and configuring eCryptfs with a trusted platform module (TPM) key

Blueprints

Installing and configuring eCryptfs with a trusted platform module (TPM) key

Note

Before using this information and the product it supports, read the information in “Notices” on page 33.

First Edition (October 2008)

Contents

Chapter 1. Scope, requirements, and support

This blueprint applies to System x® running Linux and PowerLinux™. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Systems to which this information applies

System x running Linux and PowerLinux

Intended audience

This blueprint targets Red Hat Enterprise Linux users with an intermediate level of expertise in Linux administration.

Scope and purpose

This BluePrint details the process for setting up eCryptfs to use a key sealed in a TPM version 1.2. It does not cover the details regarding setup and usage of various flavors of TPM hardware, nor does it delve into the technical details of the Trusted Computing Group (TCG) Transport Service Provider Interface (TSPI) specification.

Hardware requirements

Your equipment must have a Trusted Platform Module (TPM) version 1.2. Most currently shipping IBM® System X machines and Lenovo Thinkpad systems meet this requirement. Select IBM System P machines also have TPM devices. You must verify that you have activated the TPM in the BIOS. Refer to your system documentation for more information about managing your TPM device on your hardware.

For this guide, the reference platform is the IBM ThinkCentre 8212. If you are using a different hardware, Linux distribution, software releases, or swap configuration, certain steps will need to be adapted according to your environment.

Software requirements

The prerequisite software includes:
- Red Hat Enterprise Linux version 5.2
- "Software Development" package group:
  - Linux kernel version 2.6.26 or more recent
    http://kernel.org
  - ecryptfs-utils version 58 or more recent
    https://launchpad.net/ecryptfs
  - TrouSerS version 0.3.1 or more recent
    http://sourceforge.net/projects/trousers
  - tpm-tools version 1.3.1 or more recent
    http://sourceforge.net/projects/trousers
  - Keyutils version 1.2 or more recent
    http://people.redhat.com/~dhowells/keyutils

Since Red Hat Enterprise Linux 5.2 comes with a kernel older than the required kernel version 2.6.26, a new kernel will have to be installed for eCryptfs with TPM-managed key to work. Diverging from the stock kernel shipping with the official Linux enterprise distribution may lead to system instability. You will need to take this fact into consideration before proceeding.

Author names

Michael Halcrow

Other contributors

Monza Lui

Kersten Richter

IBM Services

Linux offers flexibility, options, and competitive total cost of ownership with a world class enterprise operating system. Community innovation integrates leading edge technologies and best practices into Linux.

IBM is a leader in the Linux community with over 600 developers in the IBM Linux Technology Center working on over 100 open source projects in the community. IBM supports Linux on all IBM servers, storage, and middleware, offering the broadest flexibility to match your business needs.

For more information about IBM and Linux, visit us at ibm.com/linux (https://www.ibm.com/linux)

IBM Support

Questions and comments regarding this documentation may be posted on the DeveloperWorks Security Blueprint Community Forum:

IBM Linux Security Blueprint Support Forum (http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1271)

The IBM developerWorks® discussion forums let you ask questions, share knowledge, ideas, and opinions about technologies and programming techniques with other developerWorks users. Use the forum content at your own risk. While IBM will attempt to provide a timely response to all postings, the use of this DeveloperWorks forum does not guarantee a response to every question that is posted, nor do we validate the answers or the code that are offered.

Typographic conventions

The following typographic conventions are used in this blueprint:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Chapter 2. Overview

Key management is traditionally the weakest point in deployed data encryption mechanisms. The vast majority of encryption solutions employ only basic password protection schemes, disregarding the “best practices” notion of multi-factor authentication. Most passwords that users can reasonably expect to memorize can be successfully attacked with straightforward algorithms running on today’s commodity computing devices.

Token devices such as Smart Cards represent a significant improvement, since access to encrypted data can then be predicated upon both “something you have” and “something you know.” However, deployment of such mechanisms is often prohibitively costly, cumbersome, and error-prone.

Laptops and workstations sold today, including popular Thinkpad products sold since 2002, include an integrated hardware device specified by the Trusted Computing Group (TCG) that regulates access to keys based on the state of the machine. The Trusted Platform Module (TPM) device measures the bootloader, kernel, and other critical components of the machine. The TPM can seal keys to a strictly defined system state and only allow the keys to be released should the machine be booted into its trusted configuration.

eCryptfs is a stacked cryptographic filesystem for Linux. eCryptfs has been upstream since kernel release 2.6.19, and it now ships as a Technology Preview in Red Hat Enterprise Linux version 5.2. eCryptfs is trivially deployable on existing storage devices and provides industrial-strength data confidentiality protection via well-weathered encryption mechanisms.

Note: eCryptfs will only protect your data-at-rest on secondary storage devices; you must employ proper physical and logical access protection to your machines while your data is accessible in volatile memory.

eCryptfs sports a flexible key module framework that allows pluggable modules to provide key management services. One such module allows eCryptfs to leverage keys sealed in the TPM of the machine.

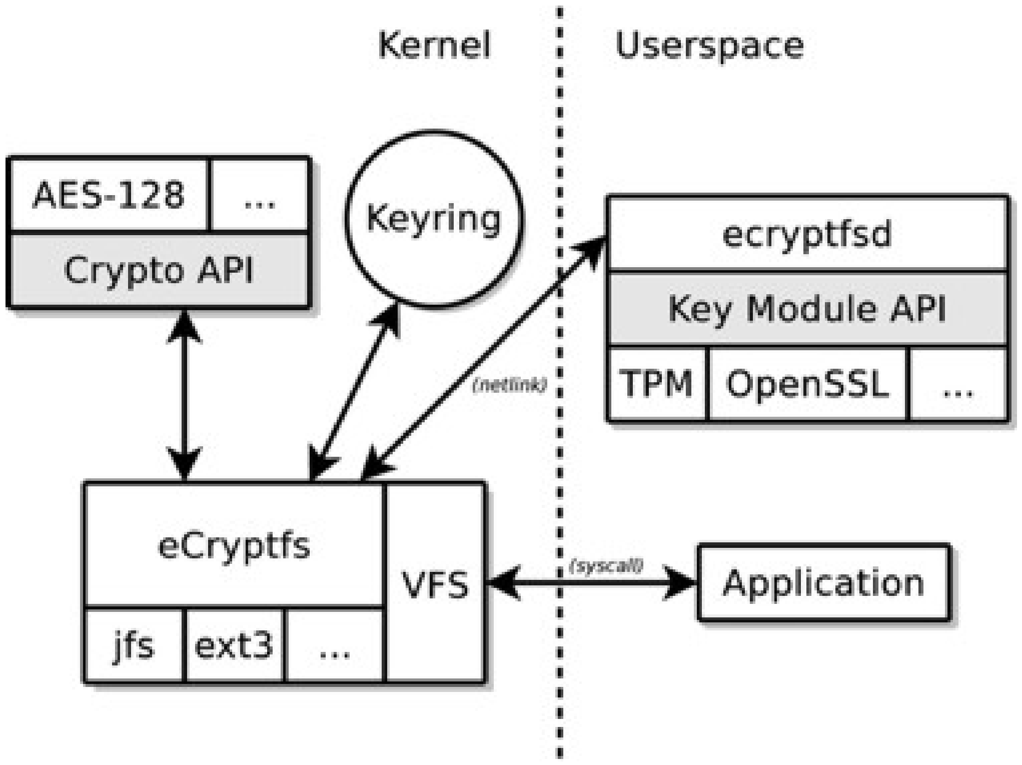

Figure 1 illustrates the eCryptfs key module framework.

eCryptfs mounts on existing directories on existing lower filesystems and encrypts on a per-file basis. Applications looking at files under the eCryptfs mount point (the stacked filesystem) will see the decrypted contents. Applications looking at the files in the lower filesystem will see the encrypted contents. This makes it trivial to securely transfer the encrypted versions of the files without requiring any modifications to applications such as incremental backup utilities.

eCryptfs uses a variety of key modules to protect the individual files. One such key module uses the TSPI (TCG Service Provider Interface). This module interfaces with TrouSerS, which is IBM's Open Source TCG Software Stack (TSS). Via this key module, the user can "seal" his files to a set of values in the Platform Configuration Registers (PCR's). The PCR values reflect the system state, in that they contain the "measurements" of critical components of the system, including the BIOS, the bootloader, the kernel, and so forth. If any of these components are not the same ones that were used to initially seal the user's keys, the TPM will refuse to unseal the key for use by eCryptfs, and the plaintext contents of the files will thus be inaccessible.

Encryption keys are locked inside the TPM, which predicates release of the keys on the system being booted into a trusted configuration. Thus, whatever authentication mechanism regulates access to the machine in its trusted state also provides access to the keys and, hence, the decrypted data. This makes access to the data only possible on a particular computing device that is booted into a particular configuration wherein the user must properly authenticate. Any existing authentication necessary to access the machine automatically and seamlessly applies to protect the data at rest.

The remainder of this document will explain how to build and install the eCryptfs software along with its dependencies, how to set up encrypted swap, how to generate a TPM-sealed key, and how to perform the eCryptfs mount with the TPM-sealed key.

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Chapter 3. Setting up eCryptfs with a TPM-managed key

Key management is traditionally the weakest point in deployed data encryption mechanisms. This document explains how to build and install the eCryptfs software along with its dependencies, how to set up encrypted swap, how to generate a TPM-sealed key, and how to perform the eCryptfs mount with the TPM-sealed key on Red Hat Enterprise Linux Version 5.2. Key tools and technologies discussed in this demonstration include eCryptfs, trusted platform module (TPM), TCG Service Provider Interface (TSPI), TrouSerS, tpm-tools, and ecryptfsd.

Related reference:

Chapter 1, "Scope, requirements, and support," on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Building and installing eCryptfs

An in-depth discussion of building and installing the Linux kernel is beyond the scope of this blueprint, but included here is an example of how to build the 2.6.26.5 kernel on a freshly installed i386 system with the **Software Development** package group initially installed.

About this task

Kernel version 2.6.26.5 can base off of the kernel configuration file shipping in Red Hat Enterprise Linux 5.2. Differences between the Red Hat Enterprise Linux 5.2 kernel configuration file and the upstream 2.6.26.5 kernel configuration file do not impact eCryptfs and TPM

This guide assumes that you are starting from a standard install of Red Hat Enterprise Linux 5.2, although most popular Linux distributions will suffice. See the Software requirements in the Chapter 1, "Scope, requirements, and support," on page 1 for the list of software that you will need.

Procedure

1. Run the following set of commands to download and build the 2.6.26.5 kernel. Accept all default values during the `make oldconfig` step.

   ```
   # cd /usr/src
   # wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.26.5.tar.bz2
   # tar xjvf linux-2.6.26.5.tar.bz2
   # cd linux-2.6.26.5
   # cp /boot/config-2.6.18-92.el5 ./.config
   # make oldconfig  (hit enter for all the questions asked)
   # make
   # make modules_install
   # make install
   ```

   Note: While this document references kernel version 2.6.26.5 as an example, you should use whichever stable kernel is most recent. The most recent stable kernel can be downloaded from http://www.kernel.org.

2. Update **/boot/grub/menu.lst** (bootloader config file) to default booting to the newly installed kernel. To do this, add a new boot entry to the **/boot/grub/menu.lst** file. Change the entry to be *default=0* to point to the correct entry; in this case, the first entry.

   ```
   default=0
   timeout=5
   splashimage=(hd0,1)/boot/grub/splash.xpm.gz
   hiddenmenu
   ```

```
title Red Hat Enterprise Linux Server (2.6.26.5)
        root (hd0,1)
        kernel /boot/vmlinuz-2.6.26.5 ro root=LABEL=/1 rhgb quiet
        initrd /boot/initrd-2.6.26.5.img
title Red Hat Enterprise Linux Server (2.6.18-92.el5)
 root (hd0,1)
 kernel /boot/vmlinuz-2.6.18-92.el5 ro root=LABEL=/ rhgb quiet
 initrd /boot/initrd-2.6.18-92.el5.img
```

3. Reboot the system and verify that the default boot setting has taken effect by running these commands:

   ```
   # reboot
   # uname -a
   ```

4. The TrouSerS package version 0.3.1-4.el5 ships as part of Red Hat Enterprise Linux 5.2. Install both the primary and the devel components of the package by running the following:

   ```
   # yum install trousers trousers-devel
   ```

   Alternatively, you can build and install the source TrouSerS package from http://sourceforge.net/projects/trousers:

   ```
   # tar xzf trousers-0.3.1.tar.gz
   # cd trousers-0.3.1
   # ./configure
   # make
   # make install
   ```

5. The tpm-tools package version 1.3.1-1.el5 ships as part of Red Hat Enterprise Linux 5.2. Install this package by running the following command:

   ```
   # yum install tpm-tools
   ```

   Alternatively, you can build and install the source tpm-tools package from http://sourceforge.net/projects/trousers:

   ```
   # tar xzvf tpm-tools-1.3.1.tar.gz
   # cd tpm-tools-1.3.1
   # ./configure
   # make
   # make install
   ```

6. When configuring ecryptfs-utils during the build process, you need to explicitly enable the TSPI key module. Download and install the latest version of ecryptfs-utils at https://launchpad.net/ecryptfs

   ```
   # wget http://downloads.sourceforge.net/ecryptfs/ecryptfs-utils-58.tar.gz
   # tar xjf ecryptfs-utils-58.tar.bz2
   # cd ecryptfs-utils-58
   # ./configure --prefix=/usr --enable-tspi
   # make
   # make install
   ```

7. Boot the machine into its trusted configuration.
8. Verify that you have loaded the tpm_tis correct driver for the TPM device.

   ```
   # modprobe tpm_tis
   ```

9. Ensure that you have started the **tcsd** daemon:

   ```
   # /etc/init.d/tcsd start
   ```

10. Take ownership of your TPM using the utility from the tpm-tools package. Do not enter any password for the Storage Root Key (SRK):

    ```
    # tpm_takeownership
    Enter owner password:
    Confirm password:
    Enter SRK password: <Hit Enter, do not input password>
    Confirm password: <Hit Enter, do not input password>
    ```

    Note: You can only take ownership once in every reset of your TPM device Prior to taking ownership, you may need to reset your TPM device in your BIOS.

a. On the ThinkCentre model 8212, enable the BIOS TPM device reset menu option by holding down the Control key on the keyboard while the system is booting from a power-off state and then entering into the BIOS setup by pressing F1 key.
b. In the BIOS setup, select **Security->TCG Feature Setup**.
c. Set **TCG Security Feature** to **Active** and the **Clear TCG Security Feature** to **Yes**.
d. Select **Save and Exit (F10)**.

   On the IBM ThinkCentre 8212, the **Clear TCG Security Feature** will not show up in the BIOS menu if the system is not powered off first and then powered back on by pressing the power button. The option will also not show up if the Control key is not held while the system is powered on.

What to do next

General TPM device management varies depending on the host BIOS and TPM chipset and is beyond the scope of this document; consult the documentation for your specific machine type to determine how to properly manage your TPM device.

Related reference:

Chapter 1, "Scope, requirements, and support," on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Setting up encrypted swap

At this stage, you should enable encryption for your swap device with a random key on boot so that your sensitive data in your system memory is not written in the clear to your swap. You can do so by creating a new boot script that runs whenever you start your machine.

About this task

To create a new boot script, follow these steps:

Procedure

1. Determine which device you have configured as your swap device by inspecting the contents of **/proc/swaps**. The contents will look like the following example. In this example, the swap device is **/dev/hda3**, and so **/dev/hda3** is subsequently used in these examples. Substitute your own swap device as you follow these steps.

   ```
   # cat /proc/swaps
   Filename   Type       Size     Used    Priority
   /dev/hda3  partition  2096472  0       -1
   ```
2. Create a new file, **/etc/init.d/encrypted-swap**, with the contents listed below. Substitute your actual swap device for **/dev/hda3** in the `PARTITION=/dev/hda3` line in the script. Ensure that the script's ownership and permissions are the same as the other scripts in **/etc/init.d**. For example, `chmod 755 /etc/init.d/encrypted-swap`.

   ```
   #!/bin/sh
   #
   # encrypted-swap Start and stop encrypted swap
   #
   # chkconfig: 2345 20 20
   # description: Start and stop encrypted swap

   PATH=/sbin:/bin
   # Substitute your actual swap device for /dev/hda3 in the line below
   PARTITION=/dev/hda3
   CIPHER=twofish-cbc-plain
   DEVICE_NAME=swap
   ```

```
. /etc/init.d/functions

case "$1" in
  start|"")
        if (cryptsetup -c $CIPHER -d /dev/random -s 128 create $DEVICE_NAME \
            $PARTITION && mkswap /dev/mapper/$DEVICE_NAME && swapon -p 1 \
            /dev/mapper/$DEVICE_NAME) >/dev/null
        then
                echo "Swap enable success"
        else
                echo "Swap enable failure"
        fi
        ;;
  stop)
        if swapoff /dev/mapper/$DEVICE_NAME && cryptsetup remove $DEVICE_NAME
        then
                echo "Swap disable success"
        else
                echo "Swap disable failure"
        fi
        ;;
  *)
        echo "Usage: $0 [start|stop]" >&2
        exit 3
        ;;
esac
exit $RETVAL
```

3. Disable your current swap device:

   ```
   # swapoff -a
   ```

4. Verify that there is no swap entry

   ```
   # swapon -s
   ```

5. Remove or comment out the swap device line from your **/etc/fstab** file. The line to delete will have the word "**swap**" in the second and third columns.
6. Enable your encrypted swap device by running the following:

   ```
   /etc/init.d/encrypted-swap start
   ```

7. Add the new script to your system startup script set:

   ```
   # chkconfig --add encrypted-swap
   ```

8. Verify that your new encrypted swap is active:

   ```
   # swapon -s
   ```

Related reference:

Chapter 1, "Scope, requirements, and support," on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Generating a TPM-sealed key

Once you have taken ownership of your TPM device and set up encrypted swap, you can now generate your own sealed key. This key effectively seals your encrypted files to a particular trusted system state.

Run `ecryptfs-generate-tpm-key` with the PCR indexes that you want to bind with your key. In this example, the TPM uses PCR indexes 0, 2, and 3 to seal the generated key:

```
# ecryptfs-generate-tpm-key -p 0 -p 2 -p 3
```

These PCR values reflect the state of various components of the system, including the BIOS, the bootloader, the kernel, and so forth. The PCR indexes that you will want to select will depend on the specific system components to which you want to "seal" or "bind" your encrypted data. In the event that one of those components changes, your encrypted data will not be decryptable. You can determine which registers change according to various components of your system by referring to the documentation for your particular machine or by changing the components, rebooting, and then checking to see which PCR registers changed as a result.

The expected output will look something like the following. The files written under these specific PCR values will only be accessible in the future when these PCR values match.

```
Success: Key created bound to:
PCR 0: 956cf5676bbc56480f541a528c6904852fbf0825
PCR 2: 24c4b4c81760a64bd3701d5928dee2128ea561f3
PCR 3: ba4e139ec3081aab583be56cab79adf584becc63
And registered in persistent storage with uuid:
0d090fc532cfe4c716ae0335f7e48be0
```

Note: Make sure to save the UUID to a safe location so that you can later find it when needed.

If you failed to generate the TPM key, see Appendix A, “Troubleshooting tips,” on page 29 section.

If you want to have a mount point accessible by a non-root user account, you may run this command as a regular user. Note the UUID returned from the key generation utility.

See The Trusted Platform Module (TPM) and How to Use It In the Enterprise PDF file for general information about the TPM device and its PCR registers (http://www.trustedcomputinggroup.org/files/temp/4B551C9F-1D09-3519-AD45C1F0B5D61714/TPM%20Overview.pdf).

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Mounting eCryptfs

Now that you have generated your sealed key, you can mount eCryptfs, telling it to use the TSPI key module and the UUID for your newly generated key.

About this task

The following will create a eCryptfs mounted new directory **/secret**.

Procedure

1. Load the eCryptfs kernel module, setting the default message wait timeout period to 1 hour to provide more than enough time for the TPM hardware to respond when the system is under heavy load:

   ```
   # modprobe ecryptfs ecryptfs_message_wait_timeout=3600
   ```

2. Run the eCryptfs daemon:

   ```
   # ecryptfsd
   ```

3. Create the directory that will house the lower encrypted files and create the mount point directory. These two directories can be the same; in this example, the directory is **/secret**:

   ```
   # mkdir /secret
   ```

Results

The following shows what to expect during the mount process. Answers to the questions are bold-typed. You will need to provide your unique UUID at this time.

```
[root@mylogin ~]# mount -t ecryptfs /secret /secret
Select key type to use for newly created files:
 1) openssl
 2) passphrase
 3) tspi
Selection: 3
tspi_uuid: 0d090fc532cfe4c716ae0335f7e48be0
Select cipher:
 1) aes: blocksize = 16; min keysize = 16; max keysize = 32 (not loaded)
 2) blowfish: blocksize = 16; min keysize = 16; max keysize = 32 (not loaded)
 3) des3_ede: blocksize = 8; min keysize = 24; max keysize = 24 (not loaded)
 4) twofish: blocksize = 16; min keysize = 16; max keysize = 32 (loaded)
 5) cast6: blocksize = 16; min keysize = 16; max keysize = 32 (not loaded)
 6) cast5: blocksize = 8; min keysize = 5; max keysize = 16 (not loaded)
Selection [aes]: <Enter>
Select key bytes:
 1) 16
 2) 32
 3) 24
Selection [16]: <Enter>
Enable plaintext passthrough (y/n) [n]:
Attempting to mount with the following options:
  ecryptfs_key_bytes=16
  ecryptfs_cipher=aes
  ecryptfs_sig=9b5665729f9b14f1
WARNING: Based on the contents of [/root/.ecryptfs/sig-cache.txt],
it looks like you have never mounted with this key
before. This could mean that you have typed your
passphrase wrong.

Would you like to proceed with the mount (yes/no)? yes
Would you like to append sig [9b5665729f9b14f1] to
[/root/.ecryptfs/sig-cache.txt]
in order to avoid this warning in the future (yes/no)? yes
Successfully appended new sig to user sig cache file
Mounted eCryptfs
```

What to do next

Once the mount has successfully completed, any files accessed under the **/secret** mount point will be protected by the key sealed in the TPM.

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Mount on boot

eCryptfs can be set up to automatically mount on boot by adding a boot script /etc/init.d/ecryptfs-tpm-mount.

In the example boot script below, substitute your specific UUID (TSPI_UUID), lower encrypted directory (SRC_PATH), and mount point (MOUNT_POINT) for those given. By specifying these parameters on the command line rather then entering them interactively, the eCryptfs mount helper will not pause during the boot to prompt for these parameters. Ensure that the script’s ownership and permissions are the same as the other scripts in /etc/init.d (e.g. chmod 755 /etc/init.d/ecryptfs-tpm-mount).

```
#!/bin/sh
#
# ecryptfs-tpm-mount Start and stop eCryptfs with TPM mount
#
# chkconfig: 2345 81 81
# description: Start and stop eCryptfs with TPM mount

PATH=/sbin:/bin:/usr/bin
TSPI_UUID=0d090fc532cfe4c716ae0335f7e48be0
SRC_PATH=/secret
MOUNT_POINT=/secret

. /etc/init.d/functions

case "$1" in
  start|"")
        /etc/init.d/tcsd start
        echo "Initializing eCryptfs TPM mountpoint"
        /sbin/modprobe ecryptfs \
ecryptfs_message_wait_timeout=3600
        sleep 5
        /usr/bin/ecryptfsd
        if (mount -t ecryptfs -o key=tspi:\
tspi_uuid=$TSPI_UUID,\
ecryptfs_cipher=aes,ecryptfs_key_bytes=16,\
ecryptfs_passthrough=n $SRC_PATH $MOUNT_POINT) >/dev/null
        then
                echo "eCryptfs mount success"
        else
                echo "eCryptfs mount failure; check the syslog for more information"
        fi
        ;;
  stop)
        if umount /secret
        then
                echo "eCryptfs umount success"
        else
                echo "eCryptfs umount failure"
        fi
        ;;
  *)
        echo "Usage: $0 [start|stop]" >&2
        exit 3
        ;;
esac
exit $RETVAL
```

As is the case with the encrypted-swap boot script, you can add the ecryptfs-tpm-mount script to the set of scripts run at system boot time with the chkconfig command:

```
# chkconfig --add ecryptfs-tpm-mount
```

On Red Hat Enterprise Linux version 5.2, the system requires additional SE Linux policy to permit eCryptfs mount-on-boot. You can ignore the following if you are not running SE Linux (i.e. sestatus outputs disabled), although we recommend keeping SE Linux enabled. To compile this policy, you need to install the selinux-policy-devel package.

```
# yum install selinux-policy-devel
```

Write this content to /root/ecryptfs.te:

```
policy_module(ecryptfs, 1)

require { type default_t; };
require { type mount_t; };
require { type sysfs_t; };
require { type user_home_dir_t; };
require { type user_home_t; };

allow mount_t default_t:dir write;
allow mount_t self:key { write search link };
allow mount_t sysfs_t:file read;
allow mount_t user_home_dir_t:dir setattr;
allow mount_t user_home_dir_t:file read;
allow mount_t user_home_t:file write;
```

Compile this new policy:

```
# cd /root
# make -f /usr/share/selinux/devel/Makefile ecryptfs.pp
```

Then load the policy:

```
# semodule -i /root/ecryptfs.pp
```

Your eCryptfs mount point will automatically become active on the next boot.

Should you continue to experience SE Linux denials in your particular configuration, you will need to extend the policy to cover your use case scenario.

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Chapter 4. Conclusion

Physical TPM devices on the market today offer relatively poor cryptographic performance, and so applications that make extensive use of the TPM, such as the current release of the eCryptfs TSPI key module, will thus be impacted.

Future releases of the eCryptfs TSPI module will employ more advanced key management techniques to help alleviate these immediate performance issues. Furthermore, vTPM work that is currently under way in the Open Source Software community addresses the general performance issues relating to TPM devices.

eCryptfs and TrouSerS provide a convenient way to leverage Trusted Computing hardware to enhance system security. By sealing your encryption key to a trusted system configuration, you can prevent your data from unauthorized access in case your storage device is lost or stolen or in case your machine is booted from unauthorized media. The user can leverage his existing authentication mechanisms used to normally access his system to prevent access to encrypted data-at-rest.

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Chapter 5. Installing and configuring eCryptfs with a trusted platform module (TPM) key

Installing and configuring eCryptfs with a trusted platform module (TPM) key

Scope, requirements, and support

This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Systems to which this information applies

System x running Linux and PowerLinux

Intended audience

This blueprint targets Red Hat Enterprise Linux users with an intermediate level of expertise in Linux administration.

Scope and purpose

This BluePrint details the process for setting up eCryptfs to use a key sealed in a TPM version 1.2. It does not cover the details regarding setup and usage of various flavors of TPM hardware, nor does it delve into the technical details of the Trusted Computing Group (TCG) Transport Service Provider Interface (TSPI) specification.

Hardware requirements

Your equipment must have a Trusted Platform Module (TPM) version 1.2. Most currently shipping IBM System X machines and Lenovo Thinkpad systems meet this requirement. Select IBM System P machines also have TPM devices. You must verify that you have activated the TPM in the BIOS. Refer to your system documentation for more information about managing your TPM device on your hardware.

For this guide, the reference platform is the IBM ThinkCentre 8212. If you are using a different hardware, Linux distribution, software releases, or swap configuration, certain steps will need to be adapted according to your environment.

Software requirements

The prerequisite software includes:

- Red Hat Enterprise Linux version 5.2
- "Software Development" package group:
  - Linux kernel version 2.6.26 or more recent
    http://kernel.org
  - ecryptfs-utils version 58 or more recent
    https://launchpad.net/ecryptfs
  - TrouSerS version 0.3.1 or more recent
    http://sourceforge.net/projects/trousers
  - tpm-tools version 1.3.1 or more recent

  http://sourceforge.net/projects/trousers
- Keyutils version 1.2 or more recent
  http://people.redhat.com/~dhowells/keyutils

Since Red Hat Enterprise Linux 5.2 comes with a kernel older than the required kernel version 2.6.26, a new kernel will have to be installed for eCryptfs with TPM-managed key to work. Diverging from the stock kernel shipping with the official Linux enterprise distribution may lead to system instability. You will need to take this fact into consideration before proceeding.

Author names

Michael Halcrow

Other contributors

Monza Lui

Kersten Richter

IBM Services

Linux offers flexibility, options, and competitive total cost of ownership with a world class enterprise operating system. Community innovation integrates leading edge technologies and best practices into Linux.

IBM is a leader in the Linux community with over 600 developers in the IBM Linux Technology Center working on over 100 open source projects in the community. IBM supports Linux on all IBM servers, storage, and middleware, offering the broadest flexibility to match your business needs.

For more information about IBM and Linux, visit us at ibm.com/linux (https://www.ibm.com/linux)

IBM Support

Questions and comments regarding this documentation may be posted on the DeveloperWorks Security Blueprint Community Forum:

IBM Linux Security Blueprint Support Forum (http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1271)

The IBM developerWorks discussion forums let you ask questions, share knowledge, ideas, and opinions about technologies and programming techniques with other developerWorks users. Use the forum content at your own risk. While IBM will attempt to provide a timely response to all postings, the use of this DeveloperWorks forum does not guarantee a response to every question that is posted, nor do we validate the answers or the code that are offered.

Typographic conventions

The following typographic conventions are used in this blueprint:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.

`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Overview

Key management is traditionally the weakest point in deployed data encryption mechanisms. The vast majority of encryption solutions employ only basic password protection schemes, disregarding the "best practices" notion of multi-factor authentication. Most passwords that users can reasonably expect to memorize can be successfully attacked with straightforward algorithms running on today's commodity computing devices.

Token devices such as Smart Cards represent a significant improvement, since access to encrypted data can then be predicated upon both "something you have" and "something you know." However, deployment of such mechanisms is often prohibitively costly, cumbersome, and error-prone.

Laptops and workstations sold today, including popular Thinkpad products sold since 2002, include an integrated hardware device specified by the Trusted Computing Group (TCG) that regulates access to keys based on the state of the machine. The Trusted Platform Module (TPM) device measures the bootloader, kernel, and other critical components of the machine. The TPM can seal keys to a strictly defined system state and only allow the keys to be released should the machine be booted into its trusted configuration.

eCryptfs is a stacked cryptographic filesystem for Linux. eCryptfs has been upstream since kernel release 2.6.19, and it now ships as a Technology Preview in Red Hat Enterprise Linux version 5.2. eCryptfs is trivially deployable on existing storage devices and provides industrial-strength data confidentiality protection via well-weathered encryption mechanisms.

Note: eCryptfs will only protect your data-at-rest on secondary storage devices; you must employ proper physical and logical access protection to your machines while your data is accessible in volatile memory.

eCryptfs sports a flexible key module framework that allows pluggable modules to provide key management services. One such module allows eCryptfs to leverage keys sealed in the TPM of the machine.

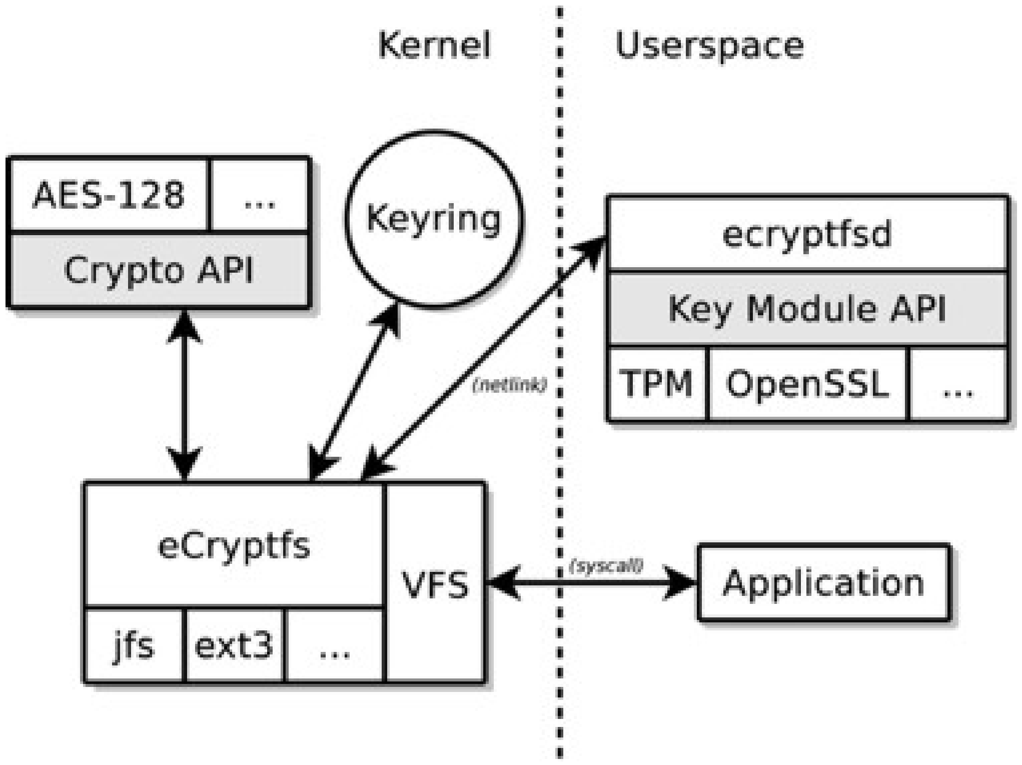

Figure 1 illustrates the eCryptfs key module framework.

eCryptfs mounts on existing directories on existing lower filesystems and encrypts on a per-file basis. Applications looking at files under the eCryptfs mount point (the stacked filesystem) will see the decrypted contents. Applications looking at the files in the lower filesystem will see the encrypted contents. This makes it trivial to securely transfer the encrypted versions of the files without requiring any modifications to applications such as incremental backup utilities.

eCryptfs uses a variety of key modules to protect the individual files. One such key module uses the TSPI (TCG Service Provider Interface). This module interfaces with TrouSerS, which is IBM's Open Source TCG Software Stack (TSS). Via this key module, the user can "seal" his files to a set of values in the Platform Configuration Registers (PCR's). The PCR values reflect the system state, in that they contain the "measurements" of critical components of the system, including the BIOS, the bootloader, the kernel, and so forth. If any of these components are not the same ones that were used to initially seal the user's keys, the TPM will refuse to unseal the key for use by eCryptfs, and the plaintext contents of the files will thus be inaccessible.

Encryption keys are locked inside the TPM, which predicates release of the keys on the system being booted into a trusted configuration. Thus, whatever authentication mechanism regulates access to the machine in its trusted state also provides access to the keys and, hence, the decrypted data. This makes access to the data only possible on a particular computing device that is booted into a particular configuration wherein the user must properly authenticate. Any existing authentication necessary to access the machine automatically and seamlessly applies to protect the data at rest.

The remainder of this document will explain how to build and install the eCryptfs software along with its dependencies, how to set up encrypted swap, how to generate a TPM-sealed key, and how to perform the eCryptfs mount with the TPM-sealed key.

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Setting up eCryptfs with a TPM-managed key

Key management is traditionally the weakest point in deployed data encryption mechanisms. This document explains how to build and install the eCryptfs software along with its dependencies, how to set up encrypted swap, how to generate a TPM-sealed key, and how to perform the eCryptfs mount with the TPM-sealed key on Red Hat Enterprise Linux Version 5.2. Key tools and technologies discussed in this demonstration include eCryptfs, trusted platform module (TPM), TCG Service Provider Interface (TSPI), TrouSerS, tpm-tools, and ecryptfsd.

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Building and installing eCryptfs

An in-depth discussion of building and installing the Linux kernel is beyond the scope of this blueprint, but included here is an example of how to build the 2.6.26.5 kernel on a freshly installed i386 system with the **Software Development** package group initially installed.

About this task

Kernel version 2.6.26.5 can base off of the kernel configuration file shipping in Red Hat Enterprise Linux 5.2. Differences between the Red Hat Enterprise Linux 5.2 kernel configuration file and the upstream 2.6.26.5 kernel configuration file do not impact eCryptfs and TPM

This guide assumes that you are starting from a standard install of Red Hat Enterprise Linux 5.2, although most popular Linux distributions will suffice. See the Software requirements in the Chapter 1, “Scope, requirements, and support,” on page 1 for the list of software that you will need.

Procedure

1. Run the following set of commands to download and build the 2.6.26.5 kernel. Accept all default values during the `make oldconfig` step.

   ```
   # cd /usr/src
   # wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.26.5.tar.bz2
   # tar xjvf linux-2.6.26.5.tar.bz2
   # cd linux-2.6.26.5
   # cp /boot/config-2.6.18-92.el5 ./.config
   # make oldconfig  (hit enter for all the questions asked)
   # make
   # make modules_install
   # make install
   ```

   Note: While this document references kernel version 2.6.26.5 as an example, you should use whichever stable kernel is most recent. The most recent stable kernel can be downloaded from http://www.kernel.org.
2. Update **/boot/grub/menu.lst** (bootloader config file) to default booting to the newly installed kernel. To do this, add a new boot entry to the **/boot/grub/menu.lst** file. Change the entry to be *default=0* to point to the correct entry; in this case, the first entry.

```
default=0
timeout=5
splashimage=(hd0,1)/boot/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.26.5)
        root (hd0,1)
        kernel /boot/vmlinuz-2.6.26.5 ro root=LABEL=/1 rhgb quiet
        initrd /boot/initrd-2.6.26.5.img
title Red Hat Enterprise Linux Server (2.6.18-92.el5)
 root (hd0,1)
 kernel /boot/vmlinuz-2.6.18-92.el5 ro root=LABEL=/ rhgb quiet
 initrd /boot/initrd-2.6.18-92.el5.img
```

3. Reboot the system and verify that the default boot setting has taken effect by running these commands:

   ```
   # reboot
   # uname -a
   ```

4. The TrouSerS package version 0.3.1-4.el5 ships as part of Red Hat Enterprise Linux 5.2. Install both the primary and the devel components of the package by running the following:

   ```
   # yum install trousers trousers-devel
   ```

   Alternatively, you can build and install the source TrouSerS package from http://sourceforge.net/projects/trousers:

   ```
   # tar xzf trousers-0.3.1.tar.gz
   # cd trousers-0.3.1
   # ./configure
   # make
   # make install
   ```

5. The tpm-tools package version 1.3.1-1.el5 ships as part of Red Hat Enterprise Linux 5.2. Install this package by running the following command:

   ```
   # yum install tpm-tools
   ```

   Alternatively, you can build and install the source tpm-tools package from http://sourceforge.net/projects/trousers:

   ```
   # tar xzvf tpm-tools-1.3.1.tar.gz
   # cd tpm-tools-1.3.1
   # ./configure
   # make
   # make install
   ```

6. When configuring ecryptfs-utils during the build process, you need to explicitly enable the TSPI key module. Download and install the latest version of ecryptfs-utils at https://launchpad.net/ecryptfs

   ```
   # wget http://downloads.sourceforge.net/ecryptfs/ecryptfs-utils-58.tar.gz
   # tar xjf ecryptfs-utils-58.tar.bz2
   # cd ecryptfs-utils-58
   # ./configure --prefix=/usr --enable-tspi
   # make
   # make install
   ```

7. Boot the machine into its trusted configuration.
8. Verify that you have loaded the tpm_tis correct driver for the TPM device.

   ```
   # modprobe tpm_tis
   ```

9. Ensure that you have started the **tcsd** daemon:

   ```
   # /etc/init.d/tcsd start
   ```

10. Take ownership of your TPM using the utility from the tpm-tools package. Do not enter any password for the Storage Root Key (SRK):

    ```
    # tpm_takeownership
    Enter owner password:
    Confirm password:
    Enter SRK password: <Hit Enter, do not input password>
    Confirm password: <Hit Enter, do not input password>
    ```

Note: You can only take ownership once in every reset of your TPM device Prior to taking ownership, you may need to reset your TPM device in your BIOS.

a. On the ThinkCentre model 8212, enable the BIOS TPM device reset menu option by holding down the Control key on the keyboard while the system is booting from a power-off state and then entering into the BIOS setup by pressing F1 key.
b. In the BIOS setup, select **Security->TCG Feature Setup**.
c. Set **TCG Security Feature** to **Active** and the **Clear TCG Security Feature** to **Yes**.
d. Select **Save and Exit (F10)**.

   On the IBM ThinkCentre 8212, the **Clear TCG Security Feature** will not show up in the BIOS menu if the system is not powered off first and then powered back on by pressing the power button. The option will also not show up if the Control key is not held while the system is powered on.

What to do next

General TPM device management varies depending on the host BIOS and TPM chipset and is beyond the scope of this document; consult the documentation for your specific machine type to determine how to properly manage your TPM device.

Related reference:

Chapter 1, "Scope, requirements, and support," on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Setting up encrypted swap

At this stage, you should enable encryption for your swap device with a random key on boot so that your sensitive data in your system memory is not written in the clear to your swap. You can do so by creating a new boot script that runs whenever you start your machine.

About this task

To create a new boot script, follow these steps:

Procedure

1. Determine which device you have configured as your swap device by inspecting the contents of **/proc/swaps**. The contents will look like the following example. In this example, the swap device is **/dev/hda3**, and so **/dev/hda3** is subsequently used in these examples. Substitute your own swap device as you follow these steps.

```
# cat /proc/swaps
Filename   Type        Size     Used    Priority
/dev/hda3  partition   2096472  0       -1
```

2. Create a new file, **/etc/init.d/encrypted-swap**, with the contents listed below. Substitute your actual swap device for **/dev/hda3** in the `PARTITION=/dev/hda3` line in the script. Ensure that the script's ownership and permissions are the same as the other scripts in **/etc/init.d**. For example, `chmod 755 /etc/init.d/encrypted-swap`.

```
#!/bin/sh
#
# encrypted-swap Start and stop encrypted swap
#
# chkconfig: 2345 20 20
# description: Start and stop encrypted swap

PATH=/sbin:/bin
# Substitute your actual swap device for /dev/hda3 in the line below
PARTITION=/dev/hda3
```

```
CIPHER=twofish-cbc-plain
DEVICE_NAME=swap

. /etc/init.d/functions

case "$1" in
  start|"")
        if (cryptsetup -c $CIPHER -d /dev/random -s 128 create $DEVICE_NAME \
            $PARTITION && mkswap /dev/mapper/$DEVICE_NAME && swapon -p 1 \
            /dev/mapper/$DEVICE_NAME) >/dev/null
        then
                echo "Swap enable success"
        else
                echo "Swap enable failure"
        fi
        ;;
  stop)
        if swapoff /dev/mapper/$DEVICE_NAME && cryptsetup remove $DEVICE_NAME
        then
                echo "Swap disable success"
        else
                echo "Swap disable failure"
        fi
        ;;
  *)
        echo "Usage: $0 [start|stop]" >&2
        exit 3
        ;;
esac
exit $RETVAL
```

3. Disable your current swap device:
   ```
   # swapoff -a
   ```
4. Verify that there is no swap entry
   ```
   # swapon -s
   ```
5. Remove or comment out the swap device line from your **/etc/fstab** file. The line to delete will have the word “**swap**” in the second and third columns.
6. Enable your encrypted swap device by running the following:
   ```
   /etc/init.d/encrypted-swap start
   ```
7. Add the new script to your system startup script set:
   ```
   # chkconfig --add encrypted-swap
   ```
8. Verify that your new encrypted swap is active:
   ```
   # swapon -s
   ```

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Generating a TPM-sealed key

Once you have taken ownership of your TPM device and set up encrypted swap, you can now generate your own sealed key. This key effectively seals your encrypted files to a particular trusted system state.

Run `ecryptfs-generate-tpm-key` with the PCR indexes that you want to bind with your key. In this example, the TPM uses PCR indexes 0, 2, and 3 to seal the generated key:

```
# ecryptfs-generate-tpm-key -p 0 -p 2 -p 3
```

These PCR values reflect the state of various components of the system, including the BIOS, the bootloader, the kernel, and so forth. The PCR indexes that you will want to select will depend on the specific system components to which you want to "seal" or "bind" your encrypted data. In the event that one of those components changes, your encrypted data will not be decryptable. You can determine which registers change according to various components of your system by referring to the documentation for your particular machine or by changing the components, rebooting, and then checking to see which PCR registers changed as a result.

The expected output will look something like the following. The files written under these specific PCR values will only be accessible in the future when these PCR values match.

```
Success: Key created bound to:
PCR 0: 956cf5676bbc56480f541a528c6904852fbf0825
PCR 2: 24c4b4c81760a64bd3701d5928dee2128ea561f3
PCR 3: ba4e139ec3081aab583be56cab79adf584becc63
And registered in persistent storage with uuid:
0d090fc532cfe4c716ae0335f7e48be0
```

Note: Make sure to save the UUID to a safe location so that you can later find it when needed.

If you failed to generate the TPM key, see Appendix A, “Troubleshooting tips,” on page 29 section.

If you want to have a mount point accessible by a non-root user account, you may run this command as a regular user. Note the UUID returned from the key generation utility.

See The Trusted Platform Module (TPM) and How to Use It In the Enterprise PDF file for general information about the TPM device and its PCR registers (http://www.trustedcomputinggroup.org/files/temp/4B551C9F-1D09-3519-AD45C1F0B5D61714/TPM%20Overview.pdf).

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Mounting eCryptfs

Now that you have generated your sealed key, you can mount eCryptfs, telling it to use the TSPI key module and the UUID for your newly generated key.

About this task

The following will create a eCryptfs mounted new directory **/secret**.

Procedure

1. Load the eCryptfs kernel module, setting the default message wait timeout period to 1 hour to provide more than enough time for the TPM hardware to respond when the system is under heavy load:
   ```
   # modprobe ecryptfs ecryptfs_message_wait_timeout=3600
   ```
2. Run the eCryptfs daemon:
   ```
   # ecryptfsd
   ```
3. Create the directory that will house the lower encrypted files and create the mount point directory. These two directories can be the same; in this example, the directory is **/secret**:
   ```
   # mkdir /secret
   ```

Results

The following shows what to expect during the mount process. Answers to the questions are bold-typed. You will need to provide your unique UUID at this time.

```
[root@mylogin ~]# mount -t ecryptfs /secret /secret
Select key type to use for newly created files:
 1) openssl
 2) passphrase
 3) tspi
Selection: 3
tspi_uuid: 0d090fc532cfe4c716ae0335f7e48be0
Select cipher:
 1) aes: blocksize = 16; min keysize = 16; max keysize = 32 (not loaded)
 2) blowfish: blocksize = 16; min keysize = 16; max keysize = 32 (not loaded)
 3) des3_ede: blocksize = 8; min keysize = 24; max keysize = 24 (not loaded)
 4) twofish: blocksize = 16; min keysize = 16; max keysize = 32 (loaded)
 5) cast6: blocksize = 16; min keysize = 16; max keysize = 32 (not loaded)
 6) cast5: blocksize = 8; min keysize = 5; max keysize = 16 (not loaded)
Selection [aes]: <Enter>
Select key bytes:
 1) 16
 2) 32
 3) 24
Selection [16]: <Enter>
Enable plaintext passthrough (y/n) [n]:
Attempting to mount with the following options:
  ecryptfs_key_bytes=16
  ecryptfs_cipher=aes
  ecryptfs_sig=9b5665729f9b14f1
WARNING: Based on the contents of [/root/.ecryptfs/sig-cache.txt],
it looks like you have never mounted with this key
before. This could mean that you have typed your
passphrase wrong.

Would you like to proceed with the mount (yes/no)? yes
Would you like to append sig [9b5665729f9b14f1] to
[/root/.ecryptfs/sig-cache.txt]
in order to avoid this warning in the future (yes/no)? yes
Successfully appended new sig to user sig cache file
Mounted eCryptfs
```

What to do next

Once the mount has successfully completed, any files accessed under the **/secret** mount point will be protected by the key sealed in the TPM.

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Mount on boot

eCryptfs can be set up to automatically mount on boot by adding a boot script /etc/init.d/ecryptfs-tpm-mount.

In the example boot script below, substitute your specific UUID (TSPI_UUID), lower encrypted directory (SRC_PATH), and mount point (MOUNT_POINT) for those given. By specifying these parameters on the command line rather then entering them interactively, the eCryptfs mount helper will not pause during the boot to prompt for these parameters. Ensure that the script’s ownership and permissions are the same as the other scripts in /etc/init.d (e.g. chmod 755 /etc/init.d/ecryptfs-tpm-mount).

```
#!/bin/sh
#
# ecryptfs-tpm-mount Start and stop eCryptfs with TPM mount
#
# chkconfig: 2345 81 81
# description: Start and stop eCryptfs with TPM mount

PATH=/sbin:/bin:/usr/bin
TSPI_UUID=0d090fc532cfe4c716ae0335f7e48be0
SRC_PATH=/secret
MOUNT_POINT=/secret

. /etc/init.d/functions

case "$1" in
  start|"")
        /etc/init.d/tcsd start
        echo "Initializing eCryptfs TPM mountpoint"
        /sbin/modprobe ecryptfs \
ecryptfs_message_wait_timeout=3600
        sleep 5
        /usr/bin/ecryptfsd
        if (mount -t ecryptfs -o key=tspi:\
tspi_uuid=$TSPI_UUID,\
ecryptfs_cipher=aes,ecryptfs_key_bytes=16,\
ecryptfs_passthrough=n $SRC_PATH $MOUNT_POINT) >/dev/null
        then
                echo "eCryptfs mount success"
        else
                echo "eCryptfs mount failure; check the syslog for more information"
        fi
        ;;
  stop)
        if umount /secret
        then
                echo "eCryptfs umount success"
        else
                echo "eCryptfs umount failure"
        fi
        ;;
  *)
        echo "Usage: $0 [start|stop]" >&2
        exit 3
        ;;
esac
exit $RETVAL
```

As is the case with the encrypted-swap boot script, you can add the ecryptfs-tpm-mount script to the set of scripts run at system boot time with the chkconfig command:

```
# chkconfig --add ecryptfs-tpm-mount
```

On Red Hat Enterprise Linux version 5.2, the system requires additional SE Linux policy to permit eCryptfs mount-on-boot. You can ignore the following if you are not running SE Linux (i.e. sestatus outputs disabled), although we recommend keeping SE Linux enabled. To compile this policy, you need to install the selinux-policy-devel package.

```
# yum install selinux-policy-devel
```

Write this content to /root/ecryptfs.te:

```
policy_module(ecryptfs, 1)

require { type default_t; };
require { type mount_t; };
require { type sysfs_t; };
require { type user_home_dir_t; };
require { type user_home_t; };

allow mount_t default_t:dir write;
allow mount_t self:key { write search link };
allow mount_t sysfs_t:file read;
allow mount_t user_home_dir_t:dir setattr;
allow mount_t user_home_dir_t:file read;
allow mount_t user_home_t:file write;
```

Compile this new policy:

```
# cd /root
# make -f /usr/share/selinux/devel/Makefile ecryptfs.pp
```

Then load the policy:

```
# semodule -i /root/ecryptfs.pp
```

Your eCryptfs mount point will automatically become active on the next boot.

Should you continue to experience SE Linux denials in your particular configuration, you will need to extend the policy to cover your use case scenario.

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Conclusion

Physical TPM devices on the market today offer relatively poor cryptographic performance, and so applications that make extensive use of the TPM, such as the current release of the eCryptfs TSPI key module, will thus be impacted.

Future releases of the eCryptfs TSPI module will employ more advanced key management techniques to help alleviate these immediate performance issues. Furthermore, vTPM work that is currently under way in the Open Source Software community addresses the general performance issues relating to TPM devices.

eCryptfs and TrouSerS provide a convenient way to leverage Trusted Computing hardware to enhance system security. By sealing your encryption key to a trusted system configuration, you can prevent your data from unauthorized access in case your storage device is lost or stolen or in case your machine is booted from unauthorized media. The user can leverage his existing authentication mechanisms used to normally access his system to prevent access to encrypted data-at-rest.

Related reference:

Chapter 1, “Scope, requirements, and support,” on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Appendix A. Troubleshooting tips

This topic discusses troubleshooting tips and caveats.

Should you run into any trouble getting eCryptfs and the TPM key module to work as expected, please visit the FAQ from the eCryptfs web site at http://ecryptfs.sourceforge.net/ecryptfs-faq.html for more information.

Encrypted Swap

If you run into trouble with your encrypted swap device, you can disable it manually by ensuring that it is turned off and that the encrypted device is disabled:

```
# swapoff /dev/mapper/swap
# cryptsetup remove swap
```

At this point, you should be able to run the /etc/init.d/encrypted-swap script to enable the encrypted swap device again.

Errors that can occur generating TPM key:

```
# ecryptfs-generate-tpm-key -p 0 -p 2 -p 3
```

Case 1: Tspi_Context_Connect failed: Communication failure

If you see this error while generating TPM key:

```
ecryptfs_generate_tpm_key.c:140: Error: Tspi_Context_Connect failed: Communication failure
```

It normally means tcsd is not running.

```
# /etc/init.d/tcsd status
```

Start the daemon by:

```
# /etc/init.d/tcsd start
```

Case 2: Tspi_Key_CreateKey failed: No SRK

If you see this error while generating TPM key:

```
ecryptfs_generate_tpm_key.c:235: Error: Tspi_Key_CreateKey failed: No SRK
```

It normally means TPM ownership has not been taken.

```
# tpm_takeownership
Enter owner password:
Confirm password:
Enter SRK password: <Enter>
Confirm password: <Enter>
```

Case 3: Tspi_Key_CreateKey failed: Authentication failed

If you see this error while generating TPM key:

```
ecryptfs_generate_tpm_key.c:235: Error: Tspi_Key_CreateKey failed: Authentication failed
```

It normally means you had entered SRK password. You will see the followings errors trying to take TPM ownership again.

```
# tpm_takeownership
Tspi_TPM_TakeOwnership failed: 0x00000008 - layer=tpm, code=0008 (8), The TPM target command has been disabled
```

If that is the case, you will have to reset TPM feature and restart from the beginning of the instructions.

The kernel options that enable eCryptfs in the 2.6.26 kernel include:

```
CONFIG_KEYS
CONFIG_CRYPTO
CONFIG_CRYPTO_ALGAPI
CONFIG_CRYPTO_AEAD
CONFIG_CRYPTO_BLKCIPHER
CONFIG_CRYPTO_HASH
CONFIG_CRYPTO_MANAGER
CONFIG_CRYPTO_CBC
CONFIG_CRYPTO_ECB
CONFIG_CRYPTO_MD5
CONFIG_CRYPTO_AES
CONFIG_ECRYPT_FS
```

Related reference:

Chapter 1, "Scope, requirements, and support," on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Appendix B. Related information and downloads

You can find additional information about the processes and tools described in these procedures.

Related information

- eCryptfs
  http://ecryptfs.sf.net
- The Linux Kernel Archives
  http://kernel.org
- eCryptfs on launchpad
  http://sourceforge.net/projects/ecryptfs
- TrouSerS on sourceforge
  http://sourceforge.net/projects/trousers
- Key utilities
  http://people.redhat.com/~dhowells/keyutils
- IBM Linux Security Blueprint Support Forum
  (http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1271)

Michael A. Halcrow, "eCryptfs: A Stacked Cryptographic Filesystem," Linux Journal Magazine (April 2007).

Related reference:

Chapter 1, "Scope, requirements, and support," on page 1
This blueprint applies to System x running Linux and PowerLinux. You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Notices

Trademarks