Linux on IBM Z and LinuxONE

Introducing
IBM Secure Execution for Linux 1.2.0

IBM

Note

Before using this document, be sure to read the information in .

Contents

Summary of changes

This revision includes maintenance and editorial changes. Technical changes or additions to the text are indicated by a vertical line to the left of the change.

What's New in September 2021

This update, SC34-7721-02, contains the following changes compared to SC34-7721-01.

New information

- Requirements for guest memory are added, see "Guest memory requirements" on page 9.
- Linux distributions that include IBM Secure Execution are now listed, see "Required software" on page 10.
- New sysfs attributes indicate whether a Linux instance detects its environment as consistent with that of a secure guest or host, see "Enabling the KVM host for IBM Secure Execution" on page 20.
- If your environment supports it, you can now use a new element in the domain-XML to simplify secure guest setup, see "Starting the secure virtual server" on page 21.
- The **genprotimg** command now has options to manage the Permit CPACF Key Management Operations (PCKMO) support, see Appendix A, "genprotimg - Generate an IBM Secure Execution image," on page 29.

What's New in December 2020

This update, SC34-7721-01, contains the following changes compared to SC34-7721-00.

New information

- Additional steps to inhibit logins from consoles are added, see "Preparing the boot image" on page 12.
- A new step that describes how to disable the virtio random-number generator is included, see "Preparing the boot image" on page 12.
- This publication now includes a troubleshooting section with tips on using debugfs, starting large guests, and attaching disks, see Chapter 7, "Troubleshooting," on page 25.

About this document

Learn about IBM Secure Execution for Linux, how you can benefit from it, how to set up a secure workload, and how the workload runs securely in a public, private, or hybrid cloud.

This publication describes IBM Secure Execution for Linux as introduced with IBM z15™ and LinuxONE III. It describes how you can create encrypted Linux images that can run on a public, private or hybrid cloud with their in-use memory protected. The publication describes how to set up the KVM host, the secure guests, and how the security works.

For details about IBM tested Linux environments, see www.ibm.com/systems/z/os/linux/resources/testedplatforms.html.

You can find the latest version of this document on IBM Documentation at:

```
https://www.ibm.com/docs/en/linux-on-systems?topic=overview-introducing-secure-execution-linux
```

Distribution independence

The information in this publication is Linux distribution independent.

Who should read this publication

For workload owners, this publication explains how IBM Secure Execution for Linux works, and what you must do to safely deploy your workload.

For cloud providers, this publication gives insights into how IBM Secure Execution for Linux works on z15™, and how the KVM host must be set up for workloads to benefit from it.

This publication assumes:

- Linux on Z or LinuxONE administrator skills.
- Familiarity with security concepts.

Terminology

IBM Secure Execution for Linux uses the terminology listed here.

boot image
    A disk image that has been prepared as a boot device. It contains all data that is required to start a Linux instance. This data includes a kernel image, an initial RAM disk, kernel parameters, and a boot loader.

host key document
    Contains the public host key in an X.509 certificate format, signed with an IBM key. A host key document is like a certificate with IBM as the trusted third party.

KVM virtual server, virtual server

    Virtualized IBM Z® resources that comprise processor, memory, and I/O capabilities as provided and managed by KVM. A virtual server can include an operating system.

KVM guest, guest, guest operating system
    An operating system of a virtual server.

KVM host, host, hypervisor
    The Linux instance that runs the KVM virtual servers and manages their resources.

protected virtualization
    An alternative name for IBM Secure Execution that still exists in some program code and, for example, in the name of the **genprotimg** command.

Other publications

These publications might be of interest.

Publications for Linux distributions

For Linux on IBM Z documents that are adapted to a particular distribution, see one of the following web pages:

- SUSE Linux Enterprise Server documents at

  ```
  ibm.com/docs/en/linux-on-systems?topic=distributions-suse-linux-enterprise-server
  ```

- Red Hat® Enterprise Linux documents at

  ```
  ibm.com/docs/en/linux-on-systems?topic=distributions-red-hat-enterprise-linux
  ```

- Ubuntu Server documents at

  ```
  ibm.com/docs/en/linux-on-systems?topic=distributions-ubuntu-server
  ```

These publications are available on IBM Documentation at ibm.com/docs/en/linux-on-systems?topic=linuxone-library-overview

- *Device Drivers, Features, and Commands*
- *Using the Dump Tools*
- *How to use FC-attached SCSI devices with Linux on z Systems®*, SC33-8413
- *Networking with RoCE Express*, SC34-7745
- *KVM Virtual Server Management*, SC34-2752
- *Configuring Crypto Express Adapters for KVM Guests*, SC34-7717
- *Introducing IBM Secure Execution for Linux*, SC34-7721
- *openCryptoki - An Open Source Implementation of PKCS #11*, SC34-7730
- *libica Programmer's Reference*, SC34-2602
- *libzpc - A Protected-Key Cryptographic Library*, SC34-7731
- *Exploiting Enterprise PKCS #11 using openCryptoki*, SC34-2713
- *Secure Key Solution with the Common Cryptographic Architecture Application Programmer's Guide*, SC33-8294
- *Pervasive Encryption for Data Volumes*, SC34-2782
- *Enterprise Key Management for Pervasive Encryption of Data Volumes*, SC34-7740
- *How to set an AES master key*, SC34-7712
- *Troubleshooting*, SC34-2612
- *Kernel Messages*, SC34-2599
- *How to Improve Performance with PAV*, SC33-8414
- *How to Set up a Terminal Server Environment on z/VM®*, SC34-2596

Dynamic Partition Manager publications

Dynamic Partition Manager (DPM) publications are available from the following web pages:

- *IBM Dynamic Partition Manager (DPM) Guide*, SB10-7170-02 available from: http://www.ibm.com/servers/resourcelink

Chapter 1. What is IBM Secure Execution?

IBM Secure Execution for Linux is a z/Architecture® security technology that is introduced with IBM z15 and LinuxONE III. It protects data of workloads that run in a KVM guest from being inspected or modified by the server environment.

In particular, no hardware administrator, no KVM code, and no KVM administrator can access the data in a guest that was started as an IBM Secure Execution guest.

Thus, IBM Secure Execution for Linux is a continuation and expansion of well-known security features of IBM Z and LinuxONE. It supplements pervasive encryption, which protects data at-rest and data in-flight, to also protect data in-use. With IBM Secure Execution for Linux, it is possible to securely deploy workloads in the cloud. The data of the workload can be protected everywhere:

- In flight with secure network protocols like TLS, SSH or IPsec
- At rest with volume encryption like dm-crypt or file system encryption like with IBM Spectrum® Scale
- In use in the memory of a running guest with IBM Secure Execution protection

When a KVM guest runs in a cloud, be it in-house or third-party, security risks to the workload include:

- Intruders who might gain root privileges due to some error in the security administration of the hypervisor.
- Malicious hypervisor code that might be introduced by exploits, including zero-day exploits, or intruders.
- Malicious virtual machines that, hypothetically, can escape the control of the hypervisor, and gain hypervisor privileges.

Intruders, malicious hypervisors, or malicious virtual machines are risks for both the cloud provider and the cloud customer, see .

To provide a secure hosting environment, a cloud provider might log every key stroke and conduct expensive audits to log any management action and deter any malicious actor.

With the introduction of pervasive encryption, all your data at rest could be encrypted with no application changes and at reasonable CPU cost.

With IBM Secure Execution, data is protected during processing. As a workload owner, your data in your KVM guest that is deployed in a cloud, which runs on IBM Z servers with IBM Secure Execution, are as safe as if you ran it in your own data center. In fact, it is safer. It is also protected from insider attacks. Only the workload owner can access the data.

Benefits of IBM Secure Execution

IBM Secure Execution provides the following benefits:

- Instead of relying on deterrence by using extensive audit tracks, IBM Secure Execution provides technology-enforced security rather than process or audit-based security.
- As a cloud provider that uses IBM Secure Execution, you can attract sensitive workloads that, formerly, were restricted to the workload owner's system.
- As a secure workload owner, you know that your workload is run in a secure manner, even outside your data center.
- As a secure workload owner, you can choose where to run your workload, independently of the security level required.
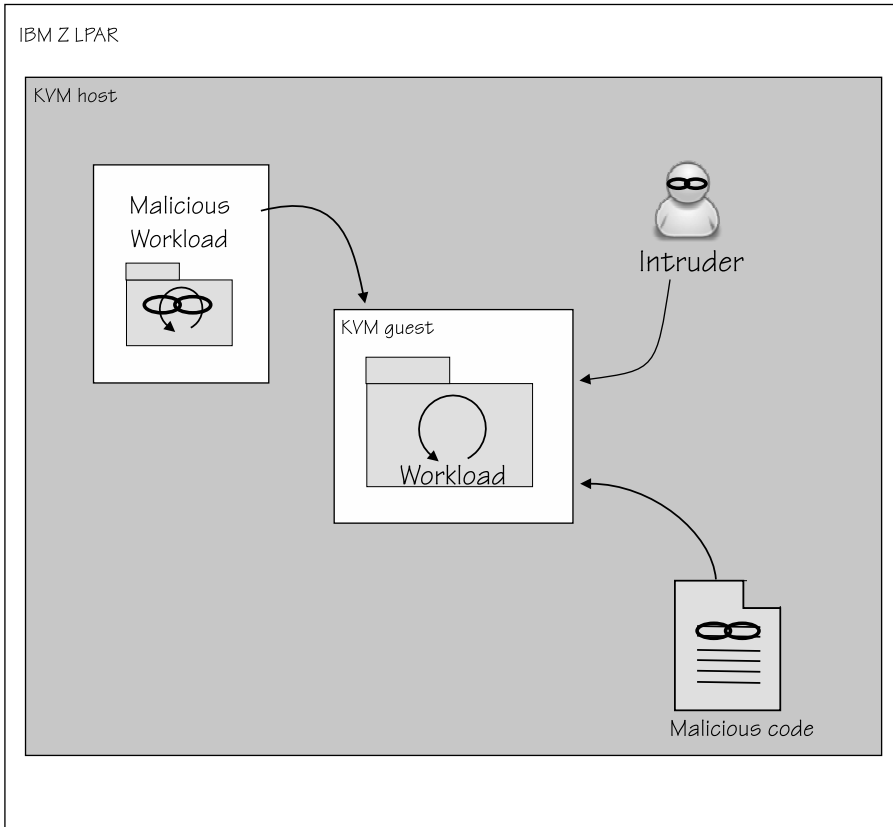
Figure 1. IBM Secure Execution protects workloads on clouds from intruders, malicious workloads, and malicious code

Chapter 2. IBM Secure Execution components

To make your workload safe in the cloud, IBM Secure Execution for Linux provides technology-based mitigation for several security threats.

With IBM Secure Execution, your workload is run in a specially protected virtual server. When you start a guest in IBM Secure Execution mode, the following aspects are protected against being observed or modified by the hosting environment:

• The boot image

• The guest memory

• The guest state

Across its entire lifecycle, such a guest has its confidentiality and integrity protected, from the moment the image is built, through the boot process and the running of the virtual server, until its termination.

IBM Secure Execution is based on hardware and firmware features that became available with the IBM z15:

• Built-in private host key

• Hardware memory protection

• Ultravisor

The ultravisor is trusted firmware that uses memory-protection hardware to enforce memory protection. The owner of a guest with IBM Secure Execution can securely pass secret information to the ultravisor by using the public host key, which is embedded in the host key document. To process the secret information, the ultravisor uses the matching private host key. The private host key is specific to an IBM Z server and is hardware protected.

A guest in IBM Secure Execution mode runs only on one or more specific, trusted servers that are provided as a cloud environment by your cloud provider. Only the workload creator can grant access to the data on it.

Boot image protection

The boot image of a secure virtual server must be prepared for the guest to run under the control of the ultravisor.

The preparation includes encrypting the boot image and computing a cryptographic hash of the image, as well as creating an IBM Secure Execution header (SE header) for this image. The header contains the image encryption keys and the hashes. The SE header itself is integrity protected, with its critical parts encrypted.

A host key document is specific to a host system on which a secure virtual server can run. A host can run the Linux instance if it can decrypt the customer root key (CRK) in the SE header. To ensure that only a particular host can run an instance, the CRK in the SE header is encrypted with that host's public key. To enable multiple hosts to run an instance, multiple encrypted copies of the CRK can be included in the SE header. Each copy is encrypted with the public key of a host where the instance can run.

Given that the boot image is encrypted, it can contain sensitive data that cannot be observed or modified by the components or operators that start the image. Typical examples of sensitive data that owners of secure virtual servers would want to place in a boot image include LUKS passphrases, root password hashes, or SSH certificates.

When the image boots, the ultravisor is given both the SE header and the encrypted image. Based on the information in the SE header the ultravisor verifies the integrity of the SE header and the image. It decrypts the image and starts the image only if all integrity checks succeed.

Memory protection

In a regular KVM setup, the KVM hypervisor can access the memory of the virtual servers. IBM Secure Execution isolates the guest memory from the KVM hypervisor by running secure virtual servers in secure memory.

Secure memory cannot be accessed from outside the secure virtual server or trusted firmware. In particular, secure memory cannot be accessed from the KVM hypervisor or any memory inspection function on the Support Element or the HMC. If a hypervisor tries to access secure memory, it gets an exception. Thus the workload is protected against data manipulation and extraction while it is running and at rest.

Once the boot image is decrypted by the ultravisor, it is stored in secure memory and all memory that is used by the secure virtual server continues to be secure.

State protection

Virtual servers need resources, such as memory and virtual CPUs to be functional. Such resources and their states are described by control blocks and comprise a large portion of the state of a virtual server. In addition, the state of a guest includes CPU registers, parts of cryptographic keys, and the program status word (PSW). Anyone who can access the state can learn something about the workload. Therefore, the ultravisor also protects all state information of a secure virtual server.

Ultravisor

To protect against control block access, IBM Secure Execution introduces a new entity, the *ultravisor*, that controls execution instead of KVM. The ultravisor decrypts the workload, secures its memory, runs, and manages it securely.

The task of the ultravisor is to load the image of a secure guest, which includes its decryption and integrity verification. It turns all memory to be used by the secure guest into secure memory and protects the state of a secure virtual server.

The ultravisor takes over all sensitive work from KVM. KVM works through a special instruction with the ultravisor. The ultravisor performs a security check on KVM's requests, and runs them, while it gives KVM only necessary information.

IBM Secure Execution protects against manipulation of the workload and tampering with memory pages. If the hypervisor needs to swap out a page, the ultravisor stores a hash of the page and encrypts it before granting the hypervisor access to the page. When the hypervisor returns the page to the guest the ultravaisor checks its contents against the stored hash.

shows three KVM guests. One guest operates in regular mode and the other two in IBM Secure Execution mode. The Ultravisor controls the IBM Secure Execution guests. The PR/SM hypervisor operates only on encrypted memory pages.

The hatched squares in the figure symbolize encrypted pages. The white squares represent memory pages that are voluntarily shared between the guests and the hypervisor to allow the hypervisor to perform I/O for the guest. The guest needs access to pages that are used to handle I/O. These pages are called bounce buffers.
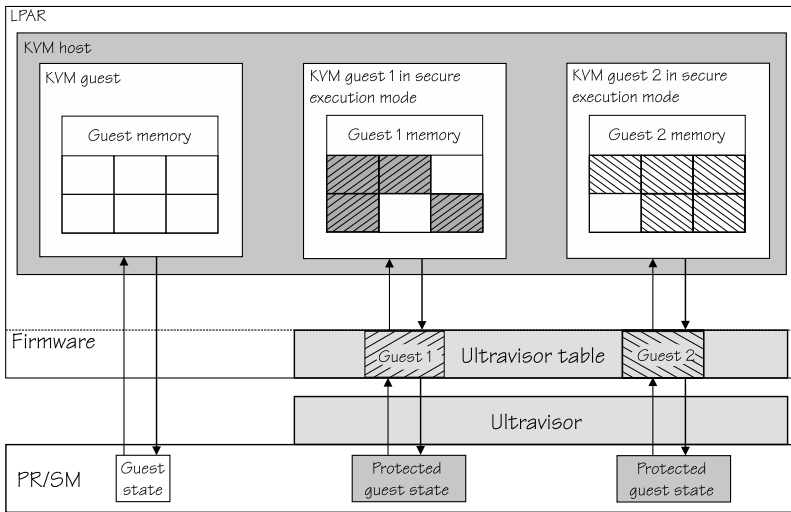
Figure 2. IBM Secure Execution protects guest memory and state

The hosting LPAR donates some memory to the ultravisor, which uses it to build a table to hold the security context data of the guest.

The secure memory that is used by a secure virtual server is labeled with the ID of the virtual server. Each virtual server can use only secure memory that is labeled with its own ID. Access to secure memory that is labeled with another ID is prevented by the IBM Z memory management hardware and firmware. This process strengthens the isolation of different guests that run in the same hypervisor. Such guest separation might be useful to cloud providers who, for example, have a legal requirement to keep workloads from different customers completely separated.

Summary

IBM Secure Execution protects the memory and state of each secure virtual server during its lifecycle.

Boot images for secure virtual servers must be encrypted and integrity protected. You use a command that automates the protection procedure. The boot image protection does not require any modification of existing applications.

It is possible to place sensitive data in the boot image that can never be observed or tampered with from the hosting environment of the secure virtual server.

A workload can only run on the hosts for which it was prepared.

Chapter 3. Securing a workload in the cloud

IBM Secure Execution encrypts the kernel image, the initial RAM file system, and the kernel parameter line. You are responsible for the application data encryption and its associated key management.

IBM Secure Execution keys

Every IBM z15 or LinuxONE III server is equipped with a private host key that is specific to that server. The key is protected by hardware and firmware. The cloud provider cannot access or manipulate the private host key. Cloud providers who run their cloud on z15 or LinuxONE III obtain a host key document from IBM. The host key document contains the public key associated with the private host key of that server. The cloud providers can distribute a host key document to cloud customers who want to run their workload in a z15 or LinuxONE III based cloud environment.

As a workload owner, you encrypt files that are necessary for booting by using the host key document of the cloud provider. The ultravisor uses the private host key, that is embedded in the hardware, to decrypt these files for the guest to boot in the cloud environment. These concepts are illustrated in
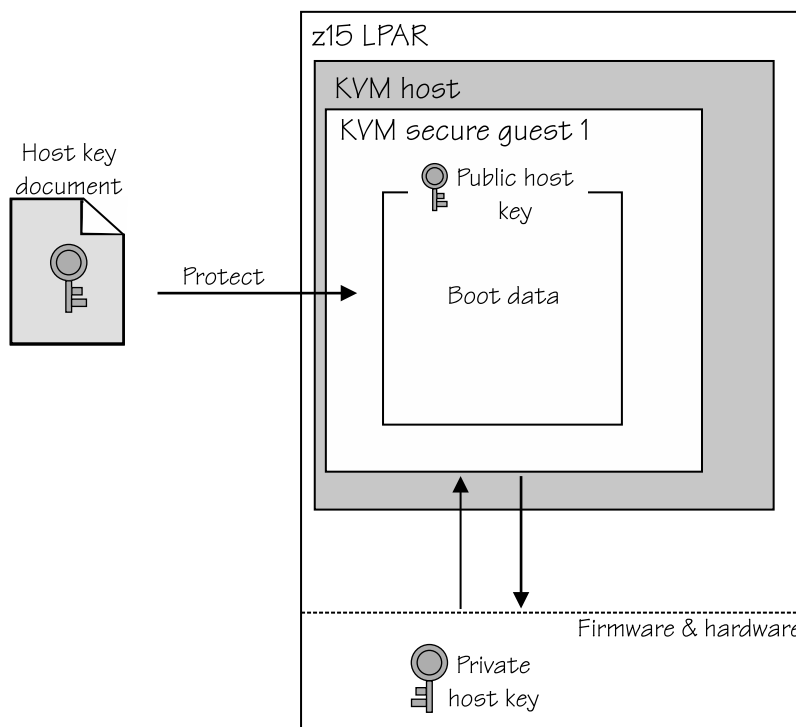

Figure 3. Encrypt files that are necessary for booting by using the host key document

Your application data is already encrypted, for example, with dm-crypt that uses LUKS volumes. This publication assumes that the data is encrypted with a symmetric key for faster encryption and decryption. The boot image that accesses the data needs access to the LUKS passphrase to use the data. Hence, you must copy the passphrase to the boot image.

IBM Secure Execution uses a cascade of encryption keys to ensure the security of the boot image. You only need to encrypt the data and use a command to secure the boot image. The cascade is shown in , and explained in the following.

Verify the host key document

Version 1.0 of IBM Secure Execution for Linux uses a manual verification procedure. Verifying the host key document is essential to ensuring the chain of trust for your workload, as shown in .
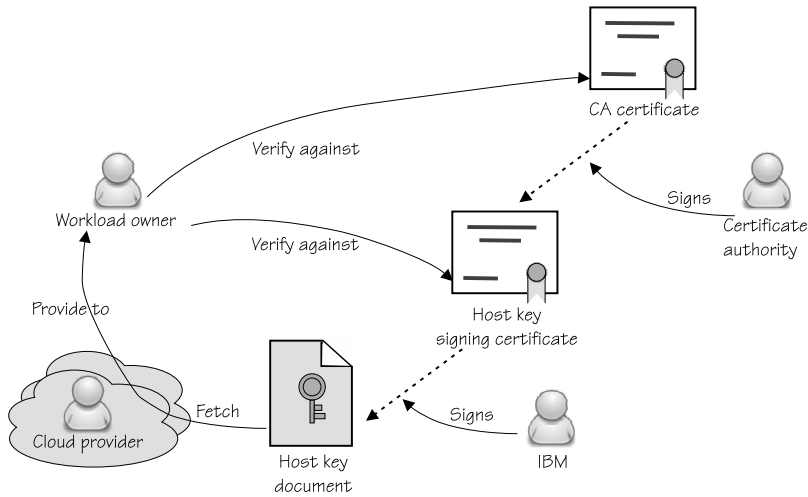
Figure 4. Chain of trust and key transfer using the host key document

Encrypt the boot image

The symmetric key that you used to encrypt your data volume would now be in the clear on the boot image. The boot image must also be encrypted with another symmetric key, the image key. To run the image, this key must be included in the IBM Secure Execution header.

Because the initial RAM file system is protected, you can in general keep secrets there, such as workload-specific keys, or network traffic protection keys.

When you secure the image header, the command you use for securing, **genprotimg**, creates an image key for you and copies it to the correct location in the IBM Secure Execution header. Optionally, you can provide your own key as an argument to the command.

Encrypt the IBM Secure Execution header

The image key, in turn, is now in the clear on the IBM Secure Execution header. To protect it, use the host key document to encrypt the IBM Secure Execution header. The only environment that can now decrypt and run the IBM Secure Execution header is the trusted hardware.

Figure 5 on page 8 shows a simplified view of the keys that are involved in all stages of securing the workload. The key used to encrypt the boot image can be automatically created and handled by the securing tool.
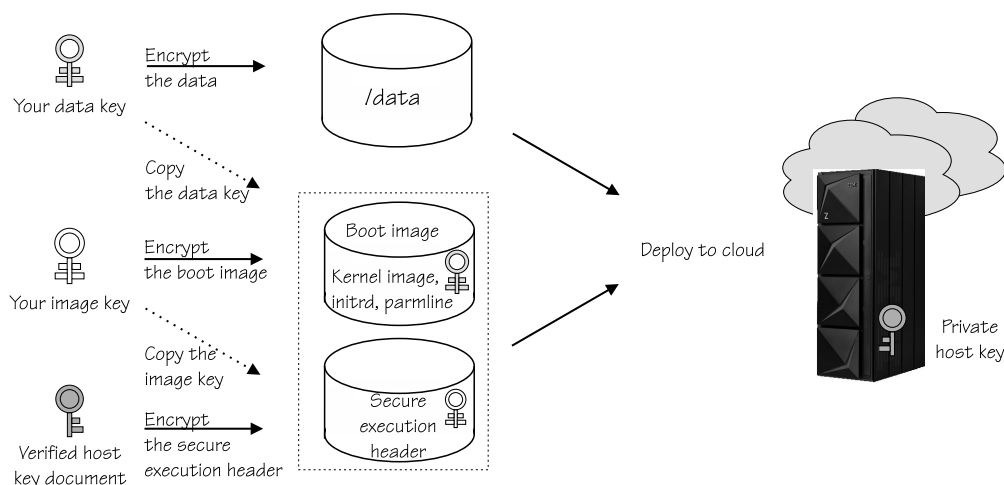

Figure 5. Conceptual key cascade for IBM Secure Execution

Chapter 4. What you should know

Before you start working with IBM Secure Execution, find out about prerequisites and restrictions.

IBM Secure Execution for Linux requires an IBM z15 or LinuxONE III with the feature installed.

As the host is not allowed to access guest memory and state, certain KVM features are not supported, including:

- Live migration. Offline migration is possible, if the guest is built for more than one host. For more information about how to build for multiple hosts, see Appendix A, "genprotimg - Generate an IBM Secure Execution image," on page 29
- Save to and restore from disk.
- Hypervisor-initiated memory dump.
- Pass-through of host devices, for example PCI, CCW, and crypto AP.
- Using huge memory pages on the host for backing guest memory.
- Memory ballooning through a virtio-balloon device.

Guest memory requirements

KVM guests in IBM Secure Execution mode require more memory than the same guests in regular mode.

The following aspects increase the memory requirement of a guest:

- The extra memory needed by a bounce buffer. Add the swiotlb value multiplied by 2 KB.
- Disk encryption, if introduced with IBM Secure Execution.
- Setting up kdump if not already in place. Double the resulting figure of the proceeding aspects.

Guests with a narrowly computed memory assignment might not boot.

Example

Consider a guest that is configured to use kdump, but no disk encryption. The swiotlb value is set to 262144. Assuming swiotlb memory blocks of 2 KB, this results in 512 MB extra memory.

For a disk that is encrypted with LUKS2, use the **cryptsetup luksDump** *<LUKS_volume>* command to display the memory needed for the key derivation function. For example (output shortened):

```
# cryptsetup luksDump /dev/vda6
LUKS header information
Version:          2
Epoch: ...

Data segments:
  0: crypt
    ...

Keyslots:
  0: luks2
    Key:       512 bits
    Priority:  normal
    Cipher:    aes-xts-plain64
    Cipher key: 512 bits
    PBKDF:     argon2i
    Time cost: 4
    Memory:    270246
    Threads:   ...
...
```

The example shows that the memory needed is 270246 KB, or approximately 271 MB. The resulting increase in guest size is:

```
2 * (512 MB + 271 MB) = 1.6 GB
```

Tip: To reduce the amount of memory that needs to be reserved for kdump, change the LUKS2 key-derivation method from the default Argon2 to PBKDF2. Use the **cryptsetup luksConvertKey** command.

Required software

Certain distributions include IBM Secure Execution.

KVM guests in IBM Secure Execution mode are supported as of these distributions:

- Red Hat Enterprise Linux 9.0 with service
- Red Hat Enterprise Linux 8.4 with service
- Red Hat Enterprise Linux 7.9 with service
- SUSE Linux Enterprise Server 15 SP3 with service
- SUSE Linux Enterprise Server 12 SP5 with service
- Ubuntu Server 20.04 LTS with service

KVM hosts in IBM Secure Execution mode are supported as of these distributions:

- Red Hat Enterprise Linux 9.0 with service
- Red Hat Enterprise Linux 8.4 with service
- SUSE Linux Enterprise Server 15 SP3 with service
- Ubuntu Server 20.04 LTS with service

Chapter 5. Workload owner tasks

As the owner of the secure workload, your tasks comprise preparing your workload and a bootable disk image that you can send to the cloud provider. Perform the steps in a trusted mainframe environment whenever possible. The steps are described as manual steps, but can be integrated into a build pipeline.

kdump consideration: If you configure kdump for the KVM guest, consider that sufficient memory must be reserved for the kdump kernel. If the memory is too small it cannot decrypt the root volume and is not functional. Configure the memory that is reserved for the crash kernel with the `crashkernel` command-line parameter.

Preparing the workload

Your goal is to prepare a workload for running as securely as possible in the cloud.

Before you begin

You require an encryption process of your choice for your data. Data here means everything except the boot image.

Important:

Do not use logical volumes together with encryption. If your distribution uses a logical volume setup by default, select a manual or expert storage setup to ensure that data is stored directly on LUKS volumes.

If logical volumes are required, use unique volume and non-predictable volume names. For example, use random names or UUIDs as generated with **uuidgen**. Multiple volumes with the same name can result in the wrong volume being mounted. With a known or easily guessed volume name, an attacker might be able to mount an unencrypted, malicious file system.

About this task

To prepare your workload for running securely in the cloud, you need to secure all parts of it. Start by securing the data volumes.

Procedure

Work in a trusted mainframe environment.

1. Prepare your data image.

   The data and the boot information can be on the same or different disk images.

   Encrypt the data partition of your disk with the encryption process of your choice.

   Tip: Use the operating system installer to encrypt the root filesystem, however, do not use the default of logical volumes with LUKS encryption, see Important note in Before you begin.

2. Ensure that the required keys and passphrases are available to the boot process.

   a) Save references to keys (plain format) or pass phrases (LUKS/LUKS2) for each volume in the `/etc/crypttab` configuration file.

   b) Include the `/etc/crypttab` configuration file in the initial RAM file system.

      Because the initial RAM file system will be encrypted, it can hold keys and pass phrases without compromising security.

Results

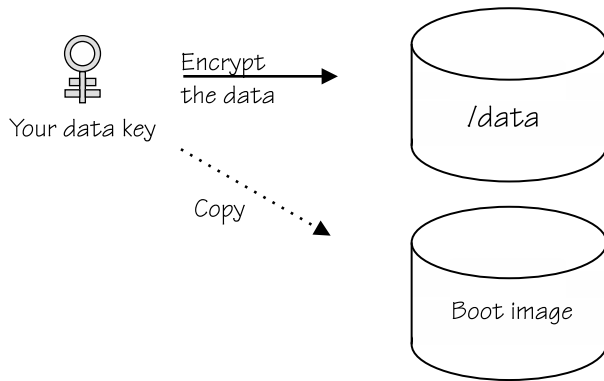As shown in Figure 6 on page 12, the workload data is encrypted, and the keys are stored in the bootable image.

Figure 6. Data volumes for a workload need to be encrypted, using, for example, pervasive encryption

What to do next
Prepare a bootable disk image, see "Preparing the boot image" on page 12.

Preparing the boot image

Prepare a KVM guest for running in IBM Secure Execution mode. The guest that you create for running in a cloud must be adequately secured. Consider all access paths to it, including console logins.

Before you begin
To prepare the guest, you need the Linux boot components:

- Kernel
- An initial RAM file system
- Kernel parameters

About this task

Your starting point is a standard KVM guest. You can use QCOW2, FCP-attached disks, or DASD disks.

Procedure

1. Install a standard Linux instance. This example uses an Ubuntu 20.04 instance.
   Accept the installer defaults, unless you want to use fixed IP addresses.
   In the package selection step, select OpenSSH to use SSH and SCP connections to your guest.
   Encrypt your root filesystem. See Important note in Preparing the boot image.
2. Prepare a kernel parameter file.
   Create a new file, called, for example `parmfile`.

   a. The boot configuration (`zipl.conf`, BLS entries, or `grub.cfg`) of the installed standard Linux instance contains a line that specifies the root device. Copy these parameters to the `parmfile`.

      Tip: Read `/proc/cmdline` to find out which parameters were used to start your Linux instance.

   b. Define a bounce buffer with the `swiotlb=` parameter.

      Tip: Use a setting of 262144 for best results.

      Add the `swiotlb=` parameter to the parameter line.

   Your `parmfile` might, for example, look like:

   ```
   root=UUID=694fd9a4-4180-4c47-92e0-7aa4fe06d370 crashkernel=196M swiotlb=262144
   ```

   You can use virt-install to set up a Linux instance:

a. Download the Ubuntu 20.04 CD-ROM image for IBM Z from http://cdimage.ubuntu.com/releases/focal/release/ into the directory `/var/lib/libvirt/images`

b. Use a command like the following to set up secguest1 as an Ubuntu 20.04 instance with 4 GB of memory on an 8 GB QCOW2 disk with the default libvirt network:

```
# virt-install --name secguest1 --memory 4096 --disk size=8 \
--cdrom /var/lib/libvirt/images/ubuntu-<version>-live-server-s390x.iso
```

Obtain the domain configuration-XML with the following command:

```
virsh dumpxml secguest1 > secguest1.xml
```

Remember to modify the XML to allow bounce buffers with `iommu="on"`.

3. Mount the directories where the kernel, the initial RAM file system, and the kernel parameter file are located.

4. Disable root login on consoles.

   • Enforce secure remote login only.

     – Set up SSHD and SSH keys.

     – Disable login on consoles by disabling serial and virtual TTYs. For example, using systemd:

```
# cat /etc/systemd/system/serial-getty@.service.d/disable.conf
[Unit]
ConditionKernelCommandLine=allowlocallogin
# cat /etc/systemd/system/autovt@.service.d/disable.conf
[Unit]
ConditionKernelCommandLine=allowlocallogin
```

     The example shows how a `disable.conf` file defines a kernel parameter, `allowlocallogin`. With this configuration file, local logins are possible if the Linux instance is started with the `allowlocallogin` kernel parameter in the parameter file that is used to build the image. Use this technique, for example, for debugging.

     – Remove information leaks on the kernel console by setting `loglevel=0` and `systemd.show_status=0`.

     – On Ubuntu Server: Edit the `/etc/securetty` to prevent console logins. Remove the contents of the file to not allow any logins. This prevents any logins from the hypervisor environment.

     For example, to remove all content in `/etc/securetty`, issue the following command:

```
# echo > /etc/securetty
```

   • Disable the debug shell in initramfs by setting the `panic=` parameter.

   • Disable debug, emergency, and rescue shells. For example, using systemd, mask the corresponding services:

```
# systemctl mask emergency.service
# systemctl mask emergency.target

# systemctl mask rescue.service
# systemctl mask rescue.target
```

5. Avoid using the virtio-rng.

   To defend against a possible malicious random-number generator on the host, exclude the virtio-rng. You can do this, for example, by using a module configuration file, `/etc/modprobe.d/virtio-rng.conf`, with the following content:

```
blacklist virtio-rng
```

6. Your guest runs in the context of a virtual server. The virtual server defines the virtual hardware. IBM Secure Execution has configuration requirements on the virtual server. See "Starting the secure virtual server" on page 21.

   Configure the QCOW2 image according to your needs. Pre-allocate it to optimize performance, or use a sparse setting to minimize size.

   For more information about the domain configuration-XML and how to configure virtual servers, see *KVM Virtual Server Management*, SC34-2752.

   Tip: Use `virt-manager` to work with the XML.

What to do next

Ensure that your guest boots and can performs its tasks, then validate the host-key document, and make the guest secure with the `genprotimg` command as described in "Securing the guest" on page 16.

Test your image

Before you secure your image and send it to the cloud provider, test it to ensure that it boots and performs its tasks as expected.

Verifying the host key document

To ensure that the host key document is genuine and provided by IBM, you need to verify the document manually.

Before you begin

To verify the host key document you need:

- The host key document that you received from your cloud provider, HKD-*<mmmm-nnnn>*.crt, or downloaded from Resource Link®, see Appendix C, "Obtaining a host key document from Resource Link," on page 35.
- The DigiCert CA certificate, `DigicertCA.crt`
- The IBM Z signing key, `ibm-z-host-key-signing.crt`
- The certificate revocation list (CRL), `ibm-z-host-key.crl`

You can download the CA certificate, the signing key, and the CRL from IBM Resource Link:

```
https://www.ibm.com/servers/resourcelink/lib03060.nsf/pages/IBM-Secure-Execution-for-Linux/
```

Check the `genprotimg` man for the latest updates to the verification procedure.

About this task

The procedure assumes an Internet connection. The commands download CRLs. For information about working offline, see the OpenSSL documentation.

Tip: Use the sample script available from s390-tools to perform the verification steps:

```
https://github.com/ibm-s390-tools/s390-tools/tree/master/genprotimg/samples/check_hostkeydoc
```

Procedure

1. Verify the CA certificate with the following command:

   ```
   # openssl verify -crl_download -crl_check DigicertCA.crt
   ```

2. Verify the signing key certificate with the following command:

```
# openssl verify -crl_download -crl_check -untrusted DigicertCA.crt ibm-z-host-key-signing.crt
```

3. Verify the signature of the host key document:

   a) Extract the public signing key into a file.
      In this example the file is called pubkey.pem:

```
# openssl x509 -in ibm-z-host-key-signing.crt -pubkey -noout > pubkey.pem
```

   b) Extract the host key signature from the host key document.
      The following command returns the offset value *<n>* of the signature:

```
# openssl asn1parse -in HKD-<mmmm-nnnn>.crt | tail -1 | cut -d : -f 1
```

      Use the resulting value *<n>* to extract the host key signature into a file called `signature`:

```
# openssl asn1parse -in HKD-<mmmm-nnnn>.crt -out signature -strparse <n> -noout
```

   c) Extract the host key document body into a file called body:

```
# openssl asn1parse -in HKD-<mmmm-nnnn>.crt -out body -strparse 4 -noout
```

   d) Verify the signature using the `signature` and body files:

```
# openssl sha512 -verify pubkey.pem -signature signature body
```

4. Verify the host key document issuer.

   Compare the outputs of the following two commands:

```
# openssl x509 -in HKD-<mmmm-nnnn>.crt -issuer -noout
```

```
# openssl x509 -in ibm-z-host-key-signing.crt -subject -noout
```

   The output of the latter MUST be:

```
subject= /C=US/ST=New York/L=Poughkeepsie/O=International Business Machines Corporation/
OU=IBM Z Host Key Signing Service/CN=International Business Machines Corporation
```

   Note: The order of the elements might differ, but it is important that all elements are present and their values are the same.

5. Verify that the host key document is still valid by checking the output of the following command:

```
# openssl x509 -in ibm-z-host-key-signing.crt -dates -noout
```

6. Verify that the host key has not been revoked.

   a) Verify the signature of the CRL file. Follow the steps described in "3" on page 15, but replace HKD-*<mmmm-nnnn>*.crt with ibm-z-host-key.crl.

   b) Verify the CRL issuer. Follow the steps described in "4" on page 15, but use the following command to find the CRL issuer:

```
# openssl crl -in ibm-z-host-key.crl -issuer -noout
```

   c) Verify that the revocation list is still valid by checking the output of the following command:

```
# openssl crl -in ibm-z-host-key.crl -lastupdate -nextupdate -noout
```

   d) Check whether the serial number of the host key is contained in the CRL.
      To find the serial number of all revoked host keys, use the following command:

```
# openssl crl -in ibm-z-host-key.crl -text -noout | grep "Serial Number"
```

To obtain the serial number of the host key document, use the following command:

```
# openssl x509  -in HKD-<mmmm-nnnn>.crt  -serial -noout
```

If the host key serial number is contained in the CRL, do not use this host key document.

Securing the guest

To convert the standard KVM guest into an IBM Secure Execution guest, run the **genprotimg** command. Also create a domain configuration-XML.

Before you begin

You require the **genprotimg** command from the s390-tools package. For more details about the **genprotimg** command, see Appendix A, "genprotimg - Generate an IBM Secure Execution image," on page 29.

The **genprotimg** command requires the following input:

- The original guest kernel
- The original initial RAM file system
- A file containing the kernel parameters
- The public host key document
- The output file name of the resulting bootable image

You must obtain the public host key document from your cloud provider. It must be available where you are preparing the guest.

Procedure

1. Ensure that you have verified the host key document, see "Verifying the host key document" on page 14.
2. The **genprotimg** command is part of the s390-tools package. If it is not already installed, download the package into the file system on your Linux instance and install it.

   For example, on an Ubuntu system, use the following command to install the s390-tools package:

   ```
   # apt install s390-tools
   ```

3. Generate the secure image.

   Run the **genprotimg** command, specifying the kernel, initial RAM disk, parameter file, host key document, and the resulting image name. Issue a command of the following form:

   ```
   # genprotimg -i <image> -r <ramdisk> -p <parm_file> \
    -k </path/to/host-key-doc>.crt -o <output_image> --no-verify
   ```

   where the host key document must match the host system for which the image is prepared. Specify multiple host key documents to enable the image to run on more than one host.

   For example:

   ```
   # genprotimg -i /boot/vmlinuz -r /boot/initrd.img -p parmfile  \
    -k HKD-8651-000201C048.crt -o /boot/secure-linux --no-verify
   ```

4. Update your boot configuration.

   a) Edit `zipl.conf`

For examples of boot configurations for different Linux distributions, see Appendix B, "Boot configurations," on page 33.

Add a new section for the IBM Secure Execution boot image and save. For example:

```
# vi zipl.conf
...
[secure]
target=/boot
image=/boot/secure-linux
...
```

Specify the location of the mounted kernel, the initramfs and the kernel parameter file directories.

b) Run **zipl -V**.

The **zipl** command creates a bootable disk image.

Results

The kernel, initial RAM file system, and parameter file are encrypted. An integrity-protected IBM Secure Execution header is created that contains all information required for booting. The IBM Secure Execution header contains the image key. The header is encrypted with the public host key.
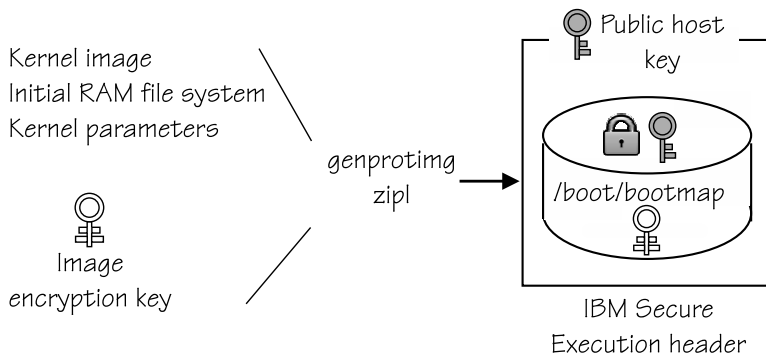

Figure 7. Boot data is consolidated and encrypted by IBM Secure Execution

What to do next

Ensure that no non-secure partitions and files on disk are included. See "Guard against non-secure partitions and files on disk" on page 17.

Guard against non-secure partitions and files on disk

You have two options for sending your encrypted boot image to the cloud provider.

Procedure

- Option 1: If you send the boot image that you created in "Securing the guest" on page 16 separately from the volume image, destroy the boot partition before you create the volume image.
  For example, if /dev/vda1 is the boot partition, use:

```
# shred /dev/vda1
```

  For information about how to boot from a separate boot image file, see Step 3 "Configure for direct kernel boot" in "Starting the secure virtual server" on page 21.

- Option 2: If you send the boot image on the unencrypted boot partition, the cloud provider can read from and write to this unencrypted partition.

  To avoid security issues, ensure that:

a) No sensitive content remains on the unencrypted partition. Use, for example, **sfill** to wipe any free space on the disk to ensure that no traces of confidential data remain. See also "Securely delete sensitive files from the unencrypted boot partition" on page 18.

b) The unencrypted partition is not mounted by the secure guest.

   Edit /etc/fstab and remove /boot (or on SUSE Enterprise Linux Server, /boot/zipl).

   With no volume mounted at /boot, or /boot/zipl, potentially non-secure new kernels or kernel updates are written to the /boot, or /boot/zipl, sub-directories of the root file system, which is backed by an encrypted volume. Because you cannot boot from an encrypted volume, these untrusted kernels are not a threat.

Securely delete sensitive files from the unencrypted boot partition

Sensitive files include the original unencrypted kernel, RAM file system, and kernel parameter file as well as the related entries in the boot configuration. These files could potentially be used by an attacker to obtain secrets.

Use, for example, the **shred** command to remove these files. Then re-run the boot configuration update.

What to do next

Supply your setup details to your provider, see "Communicating your setup to the provider" on page 18. Transfer the secure disk image and domain configuration-XML to the IBM Secure Execution host.

Communicating your setup to the provider

The cloud provider needs to know your expected disk layout and other information.

Configuration data

Your cloud provider needs to know the amount of disk storage, CPU, and network interfaces of the configuration your workload expects. Include the domain configuration-XML that you created when you send your image to the provider.

Assuming one disk, you need to specify, for example, these items:

Item	Value
Disk	Size and type
Memory	Amount:
No. of network interfaces	
Optional: MAC address	

An example is shown here:

Item	Value
Disk	Size and type: 8 GB, QCOW2
Memory	Amount: 4 MB
No. of network interfaces	2
MAC addresses	52:54:00:36:d3:75, 52:54:00:36:d3:88

Chapter 6. Cloud provider tasks

As a cloud provider, your tasks comprise setting up the KVM host and running the workload provided to you by a customer.

Before you start

You require a z15 or later mainframe with the IBM Secure Execution technology enabled and the IBM provided key bundles applied.

For information about enabling IBM Secure Execution, see *IBM Dynamic Partition Manager (DPM) Guide*, SB10-7170-02.

For how to install a key bundle, see "Importing key bundles" on page 19.

Tip: For setups with many or large guests, a large value for `vmalloc`, for example `vmalloc=1T`, might help to ensure that enough virtual contiguous host memory is available for guest addressing. Increasing the value does not cause more memory to be used.

Importing key bundles

After ordering and installing the IBM Secure Execution for Linux feature you can import key bundles, if none are installed.

About this task

You need to import keys only once, because keys are reimported with any initial microcode load (IML). To import a key bundle, follow these steps:

Procedure

1. Log in to the SE or HMC using an ID with sufficient permissions.
2. Open the **System Details** task to the Instance Information tab
3. Click the Manage button next to **Secure Execution for Linux**
4. On the **Manage Secure Execution Keys** panel, if **Host key** shows Not Installed, click the **Update** button. See Figure 8 on page 20.
   Do not change the bundle file names.

   You can import one key bundle at a time, from either an FTP server or a removable-media device.
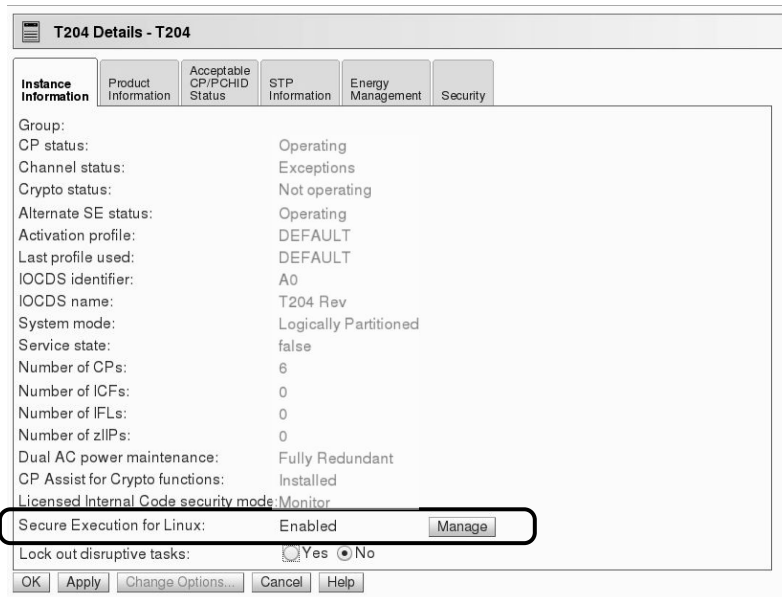
Figure 8. HMC showing how to import a key bundle

Enabling the KVM host for IBM Secure Execution

A cloud provider sets up a KVM host in an LPAR for IBM Secure Execution.

About this task

The KVM host must opt in to IBM Secure Execution, that is, to use the Ultravisor. Use the `prot_virt` kernel parameter to opt in for IBM Secure Execution on the host.

Procedure

Modify a Linux instance to be able to act as an IBM Secure Execution host.

1. Modify the boot configuration.

   For example, if you use **zipl**, add the parameter `prot_virt` to the parameters in the `zipl.conf` file and save.

   For example:

   ```
   # vi zipl.conf
   ...
   parameters=" ...prot_virt=1"
   ...
   ```

   Run **zipl**. For more information about **zipl**, see the *Device Drivers, Features, and Commands* or the man page.

2. From the HMC, IPL the device, which then boots the secure KVM host.

   The KVM host then donates some memory to the ultravisor. The ultravisor uses the memory to store the security context for memory in the LPAR. Because of this memory donation, the KVM host sees slightly less memory than what is available in the LPAR. The resulting setup is shown in .
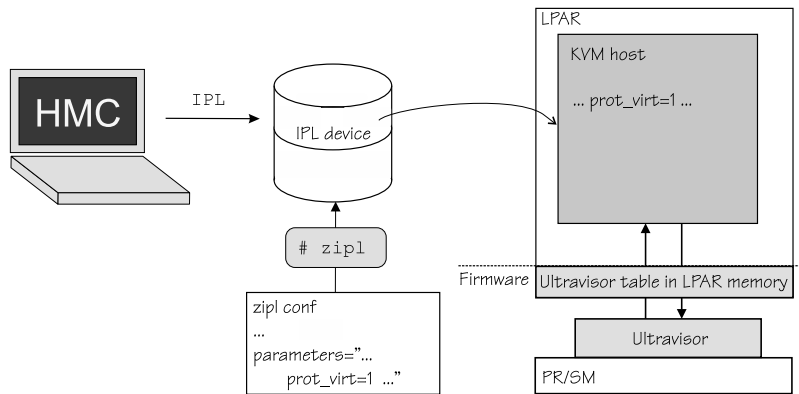
Figure 9. A KVM host is set up to run in IBM Secure Execution mode

3. Verify that the opt-in was successful.

   Check the output of the **dmesg** command. The command must show that memory was reserved for the ultravisor.

   For example:

   ```
   [ 1.010810] Reserving 322MB as ultravisor base storage
   ```

   The exact amount varies with the size of the LPAR.

   Tip: In a trusted environment, if your distribution supports it, you can read sysfs attributes as indicators of IBM Secure Execution mode:

   - `sys/firmware/uv/prot_virt_host`. The KVM host runs in IBM Secure Execution mode if the value is 1.
   - `sys/firmware/uv/prot_virt_guest`. The KVM guest runs in IBM Secure Execution mode if the value is 1.

   Note that this does not constitute full proof that the host or guest is secure.

Starting the secure virtual server

On the KVM host, create a domain configuration-XML for the virtual machines that are to run in IBM Secure Execution mode.

Before you begin

You need a bootable disk image that is encrypted with the public host key of the mainframe on which you want to run it. See "Preparing the boot image" on page 12.

Procedure

1. Place the bootable disk image on the KVM host file system in `/var/lib/libvirt/images`
   For example, assuming that the image is called `secguest1.img`:

   ```
   # ls /var/lib/libvirt/images
   ...
   secguest1.img
   ...
   ```

2. Modify the domain configuration-XML you received from your customer.

   Add the launchSecurity element with type s390-pv to set defaults that simplify configuring the virtual server for IBM Secure Execution for Linux.

   a) Optional: Confirm that this setting is available in your environment

Look for the following line in the output of the **virsh domcapabilities** command:

```
<s390-pv supported="yes">
```

b) Add the launchSecurity element.
   For example:

```
<domain type="kvm">
    ...
    <launchSecurity type="s390-pv"/>
    ...
</domain>
```

For example, this setting makes the required bounce buffer for virtio devices the default and you do not have to specify it explicitly for each device. This setting also leads to warning messages if the CPU model of the virtual server does not include all features that are required by IBM Secure Execution for Linux.

Manual domain-XML configuration

If the output of the **virsh domcapabilities** command shows that you do not have support for the launchSecurity element, you must configure the domain XML manually:

a. Ensure that the XML has `iommu="on"` set to allow the use of bounce buffers on every element that represents a virtio device, for example, the `<disk>`, `<serial>`, and `<interface>` elements.

b. Do not define a memory balloon device for secure guests. Use the following definition in the guest XML:

```
<memballoon model='none'/>
```

For example, the following domain configuration-XML, called `secguest1.xml`, configures a virtual machine called secguest1 that allows bounce buffers:

```
<domain type="kvm">
   <name>secguest1</name>
   ...
   <devices>
      <disk type="file" device="disk">
         <driver name="qemu" type="raw" cache="none" io="native" iommu="on"/>
         <source file="/var/lib/libvirt/images/secguest1.img"/>
         <target dev="vda" bus="virtio"/>
         <address type="ccw" cssid="0xfe" ssid="0x0" devno="0x1108"/>
         <boot order="1"/>
      </disk>
      ...
      <memballoon model='none'/>
   </devices>
```

For details about the domain configuration-XML and how to configure virtual servers, see *KVM Virtual Server Management*, SC34-2752.

Tip: Use **virt-manager** to work with the XML.

3. Optional: Configure for direct kernel boot.

If you received the secure boot image as a separate bootable kernel image file, modify the domain XML for a direct kernel boot.

For example, this domain XML configures a guest that is booted from a kernel image:

```
<os>
    ...
    <kernel>/var/lib/images/secure_img</kernel>
</os>
```

The `<kernel>` entry must contain the fully qualified path and file name of the secure boot image file.

4. On the KVM host console, define the virtual machine with the **virsh define** command.
   For example, to define secguest1 defined by the secguest1.xml:

```
# virsh define secguest1.xml
```

5. From the KVM host console, verify that the guest can be started with the **virsh start** command. For example, to start secguest1:

```
# virsh start secguest1
```

Results

The KVM guest defined by secguest1.img starts running in IBM Secure Execution mode. For information about troubleshooting, see "Starting virtual server fails" on page 27.

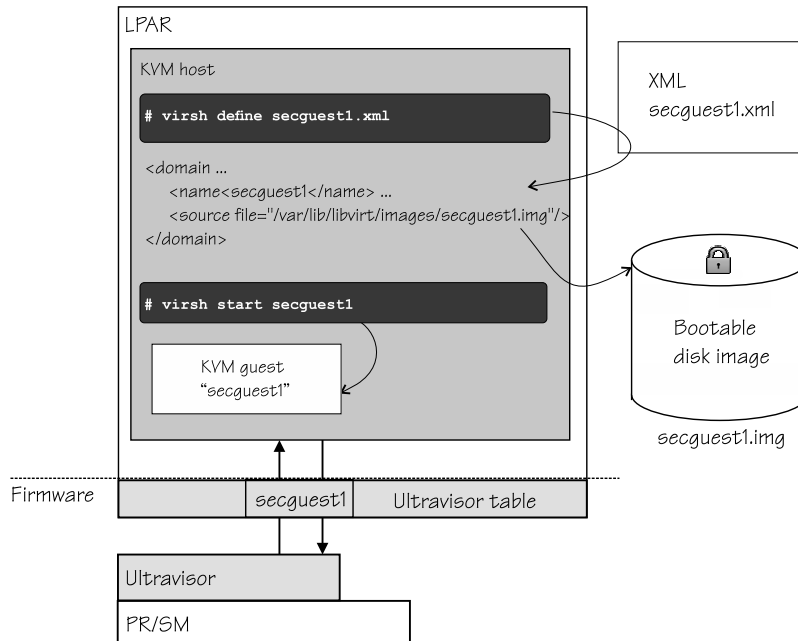*Figure 10. A KVM guest is created from a bootable image to run in IBM Secure Execution mode*

Chapter 7. Troubleshooting

Methods of problem determination include examining logs, configuring the virtual server, and using debugfs.

Procedure

Actions that you can take to help with troubleshooting include the following:

- Check that the Linux distribution you are using supports IBM Secure Execution.

  The guest console output reports errors due to a missing facility if the distribution is not current enough.
- Configure the virtual machine to preserve debug data.

  Use the on_crash element in the domain configuration XML, for example:

  ```
  <on_crash>preserve</on_crash>
  ```

- On the host, verify that the opt-in to IBM Secure Execution was successful.

  Examine the dmesg for output similar to this example:

  ```
  "[    0.828273] prot_virt: Reserving 322MB as ultravisor base storage"
  ```

  The message shows that the Ultravisor successfully reserved memory and started.
- Examine the output of the QEMU process. The output is saved in /var/log/libvirt/qemu/ *<domain>*.log. Problems might be mentioned here, for example, if a virtio device was configured without bounce buffers.
- Check the libvirt log for messages. By default, libvirt log messages are stored in the system journal. You can specify a different location in the libvirt configuration file at /etc/libvirt/ libvirtd.conf. For more information, see libvirt.org/logging.html.

  For how to set the logging level, see *KVM Virtual Server Management*, SC34-2752.
- Use debugfs, if supported.
  1. Mount debugfs with

     ```
     mount -t debugfs none /sys/kernel/debug/
     ```

  2. Find information relevant to IBM Secure Execution under

     ```
     /sys/kernel/debug/s390dbf/kvm-uv/*
     ```

- If possible, include the output from around the time of failure of the following files with your problem report:
  – /sys/kernel/debug/s390dbf/kvm-trace/sprintf
  – /sys/kernel/debug/s390dbf/kvm-uv/sprintf

Starting guests fail with error: "Protected boot has failed: 0xa02"

Unable to start IBM Secure Execution guests on a KVM host with numerous, or large guests.

Symptoms
On a KVM host with many guests defined, not all guests can start.

Causes

The kernel has run out of virtual address space to map the areas needed for the guest metadata. The size of the metadata depends on the logical size of the guest, and needs to be allocated fully in the virtual address space of the kernel.

Resolving the problem

Use the vmalloc parameter to add more virtual contiguous host memory for the addressing.

User response: Add vmalloc with a large value to the kernel command line of the KVM host, for example:

```
vmalloc=1T
```

Note: Increasing the `vmalloc` value does not cause more memory to be used.

Attaching a disk with virsh attach-disk causes guest to crash

When attempting to add a disk to a running guest using virsh attach-disk, the virtual machine reports successful attachment.

Symptoms

A **`virsh attach-disk`** command results in a success message and a subsequent crash of a guest in secure execution mode.

Causes

Guests in secure execution mode require `iommu="on"` to be set for all virtio devices to allow the use of bounce buffers. Adding a device without allowing bounce buffers destabilizes the guest.

Resolving the problem

All virtio devices must be configured with the specification `iommu="on"` to allow the use of bounce buffers.

User response: To attach a disk to a guest running in secure execution mode, use the **`virsh attach-device`** command with a device configuration XML. For details about the command, see *KVM Virtual Server Management*, SC34-2752. For an example, see "Example: Adding a disk to a guest running in secure execution mode" on page 26

Example: Adding a disk to a guest running in secure execution mode

Procedure

1. Configure the disk in a separate device configuration XML file. This configuration must include the `iommu="on"` setting.

   Specify `iommu="on"` in the XML file as an attribute of the driver element within the disk configuration.

   For example, in this `MyDisk.xml` file, iommu is set to "on" for a disk:

   ```
   <disk type="block" device="disk">
       <driver name="qemu" type="raw" cache="none" io="native"
        iothread="2" iommu="on"/>
   ...
   ```

2. Specify this device configuration XML file as an argument for the **`virsh attach-device`** command when you hotplug the disk.

   For example, this command picks up the `MyDisk.xml` configuration file for the disk:

   ```
   # virsh attach-device --config Guest1 ~/MyDisk.xml
   ```

   Tip: Also use the **`virsh attach-device`** command to attach interfaces.

Starting virtual server fails

Unable to start virtual server.

Symptoms

The virtual server cannot start.

Causes

The virtual server might not have enough memory to unlock volumes during the boot process.

Resolving the problem

Increase the memory allocated for the virtual server.

User response: Try these options:

- Increase the memory by using the `<memory>` element in the domain-configuration XML of the virtual server.
- Change the LUKS2 key-derivation method from the default Argon2 to PBKDF2 by using the **`cryptsetup luksConvertKey`** command.

Appendix A. genprotimg - Generate an IBM Secure Execution image

The **genprotimg** command builds an encrypted boot record from a given kernel, initial RAM disk, parameters, and public host-key document.

Command availability

The command is included on Linux distributions as of:

- Red Hat Enterprise Linux 8.3
- Ubuntu Server 20.04
- SUSE Linux Enterprise Server 15.2

If your distribution does not contain the **genprotimg** command, you can either copy the kernel and initial RAM file to a an environment that includes genprotimg and build the secure image there, or build the command yourself from the source on github:

```
https://github.com/ibm-s390-tools/s390-tools/tree/master/genprotimg
```

genprotimg syntax
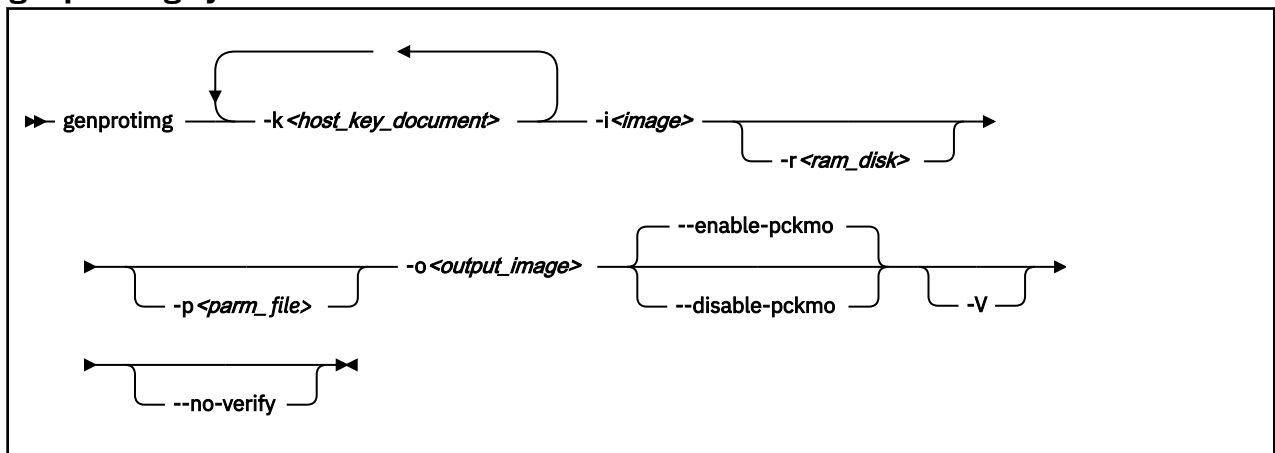
Parameters

-k *<host_key_document>* **or --host-key-document=***<host_key_document>*
　　Specifies the host key document. The document must match the host system for which the image is prepared. Specify multiple host key documents to enable the image to run on more than one host. The document is a plain text file with a name of the form: HKD-*<type>*-*<serial>*.crt

-i *<image>* **or --image=***<image>*
　　Specifies the Linux kernel image.

　　Note: The **genprotimg** command cannot use an ELF file as a Linux kernel image.

-r *<ramdisk>* **or --ramdisk=***<ramdisk>*
　　Specifies a RAM file system.

-p *<parm_file>***or --parmfile=***<parm_file>*
　　Provides a file with kernel parameters.

-o or --output
　　Specifies the target image name.

--disable-pckmo

Disables the Permit CPACF Key Management Operations (PCKMO) support.

The PCKMO options configure key management operations on the virtual server. If enabled, keys can be created using the DEA, TDEA, AES, or ECC algorithms.

--enable-pckmo

Enables the PCKMO support. This is the default.

Interface change:

For genprotimg versions with the `--enable-pckmo` option, PCKMO key operations are enabled by default. To confirm that `--enable-pckmo` is available on your distribution, issue:

```
# genprotimg -h
```

If the `--enable-pckmo` option is listed, no further action is needed to enable PCKMO operations. To return to the previous behavior, specify `--disable-pckmo`.

If no `--enable-pckmo` option is listed, and you want PCKMO operations, try:

```
# genprotimg ... --x-pcf '0xe0'
```

-V or --verbose

Prints additional runtime information.

--no-verify

Specifies that the host key document is not verified.

> ⚠️ **Warning:** As long as a manual procedure (see "Verifying the host key document" on page 14) is in place for verification, use the `--no-verify` option. Working with an unverified key makes your image vulnerable to man-in-the-middle attacks. Whoever gave you the host key document might be able to decrypt your image.

-v or --version

Displays the version information for the command.

-h or --help

Displays out a short help text, then exits. To view the man page, enter **man genprotimg**.

--help-experimental

Displays experimental usage information, then exits.

--help-all

Displays all help text, then exits.

Example: Using genprotimg to generate an IBM Secure Execution image

Assume that you have an Ubuntu guest that you would like to convert into an IBM Secure Execution guest. You have the following information ready:

- The guest has the following `zipl.conf`:

```
[ubuntu]
target=/boot
image=/boot/vmlinuz
ramdisk=/boot/initrd.img
parameters=root=UUID=694fd9a4-4180-4c47-92e0-7aa4fe06d370 crashkernel=196M
```

- A host key document called `HKD-8651-00020089A8.crt`,

1. Verify the host key document, see "Verifying the host key document" on page 14.

2. Create a parameter file called `parmfile`. Copy the content of the parameter that specifies the root device.

3. Specify bounce buffers with a `swiotlb` parameter with a value of `262144`.

The result is a parameter file with the following content:

```
root=UUID=694fd9a4-4180-4c47-92e0-7aa4fe06d370 crashkernel=196M swiotlb=262144
```

4. Generate an IBM Secure Execution image in /boot/secure-linux with the command:

```
# genprotimg -i /boot/vmlinuz -r /boot/initrd.img -p parmfile --no-verify
-k HKD-8651-00020089A8.crt -o /boot/secure-linux
```

Appendix B. Boot configurations

By default, **zipl** processes the default configuration in the default configuration file `/etc/zipl.conf`.

A generic zipl configuration file

```
# vi zipl.conf
...
[secure]
target=/boot
image=/boot/secure-linux
...
```

Red Hat Enterprise Linux BLS configuration

You can use the `zipl.conf` configuration file or BLS snippets to configure the booting of a Red Hat Enterprise Linux guest. A sample BLS snippet could look as follows:

```
title Red Hat Enterprise Linux (secured)
version 5.5.0-10-bls-test
linux /boot/secure-linux
```

Assuming that this zipl configuration-file and the BLS snippet are both at their default locations, the following command processes the BLS snippet:

```
# zipl -V
```

SUSE Linux Enterprise Server GRUB2 configuration

This example assumes that the `/boot` filesystem resides on a BRTFS volume.

1. Append the following text to `/etc/grub.d/40_custom`:

   ```
   menuentry "Secure execution image" {
       linux ${btrfs_subvol}/boot/secure-linux
   }
   ```

2. Run the following command:

   ```
   # grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

For more information on GRUB2, see the SUSE Linux *Administration Guide,* available at:

```
https://documentation.suse.com/sles
```

Ubuntu Server zipl configuration

The following sample zipl.conf file shows a setup for an Ubuntu Linux instance:

```
[ubuntu]
target=/boot
image=/boot/secure-linux
```

Assuming that this zipl configuration file is at its default location, the following command processes the Ubuntu definition:

```
# zipl -V
```

Appendix C. Obtaining a host key document from Resource Link

You can download a host key document from Resource Link, if the IBM Secure Execution feature is enabled on your IBM Z or LinuxONE.

Before you begin

Resource Link offers a search page for host key documents. If you have never signed in to Resource Link, you need to register before you can access the host key document search page.

1. Open the main page, www.ibm.com/servers/resourcelink
2. Click **Sign in**. You are prompted to register.

You will receive an email in about an hour when the registration is complete.

Procedure

1. As a registered user, access the search page:

   ```
   https://www.ibm.com/servers/resourcelink/hom03010.nsf/pages/HKDSearch?OpenDocument
   ```

2. Select the machine type and enter the machine serial number.

   If an HKD file is available for the machine, a link is displayed to download the file, as on the page shown in Figure 11 on page 35.

Figure 11. Host key document search page on Resource Link

3. Click the link to download the document.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at
www.ibm.com/legal/copytrade.shtml

Adobe is either a registered trademark or trademark of Adobe Systems Incorporated in the United States, and/or other countries.

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Red Hat® is a trademark or registered trademark of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Index

IBM®