Linux on Z and LinuxONE

Introducing IBM Secure Execution for Linux 1.1.0

SC34-7721-00

Note

Before using this document, be sure to read the information in “Notices” on page 29.

This edition applies to IBM® Secure Execution for Linux® 1.1.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this document v
Who should read this publication v
Terminology v
Other publications vi

Chapter 1. What is IBM Secure Execution? 1

Chapter 2. IBM Secure Execution components 3

Chapter 3. Securing a workload in the cloud 7

Chapter 4. What you should know 9

Chapter 5. Workload owner tasks 11
Preparing the workload 11
Preparing the boot image 12
Verifying the host key document 14
Securing the guest 16
Communicating your setup to the provider 17

Chapter 6. Cloud provider tasks 19
Importing key bundles 19
Enabling the KVM host for IBM Secure Execution 20
Starting the secure virtual server 21

Appendix A. genprotimg - Generate an IBM Secure Execution image 23

Appendix B. Boot configurations 25

Appendix C. Obtaining a host key document from Resource Link 27

Notices 29
Trademarks 29

Index 31

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 the IBM Knowledge Center at:

```
https://www.ibm.com/support/knowledgecenter/en/linuxonibm/liaaf/lnz_r_dse.html
```

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 works, and what you must do to safely deploy your workload.

For cloud providers, this publication gives insights into how IBM Secure Execution 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 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 Z documents that are adapted to a particular distribution, see one of the following web pages:

- SUSE Linux Enterprise Server documents at

```
www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_suse.html
```

- Red Hat Enterprise Linux documents at

```
www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_rh.html
```

- Ubuntu Server documents at

```
www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_ubuntu.html
```

These publications are available on IBM Knowledge Center at www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_lib.html

- *Device Drivers, Features, and Commands*
- *Using the Dump Tools*
- *KVM Virtual Server Quick Start*, SC34-2753
- *KVM Virtual Server Management*, SC34-2752
- *How to use FC-attached SCSI devices with Linux on z Systems®*, SC33-8413
- *Introducing IBM Secure Execution for Linux*, SC34-7721
- *libica Programmer's Reference*, SC34-2602
- *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
- *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 that 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 Figure 1 on page 2.

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.

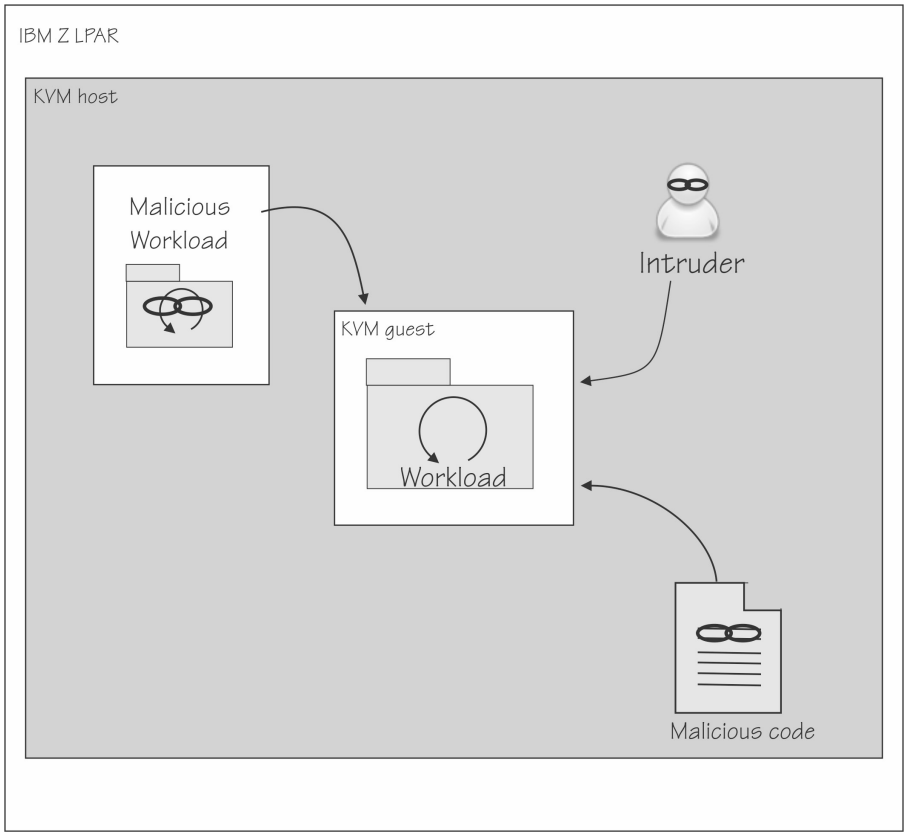


Figure 1. IBM Secure Execution protects workloads on clouds from intruders, malicious workloads, and malicious code

Chapter 2. IBM Secure Execution components

To make your workload safe in the cloud, IBM Secure Execution 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 with IBM Secure Execution, 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 makes use of 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 with IBM Secure Execution runs only on one or more specific, trusted machines 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.

Figure 2 on page 5 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.

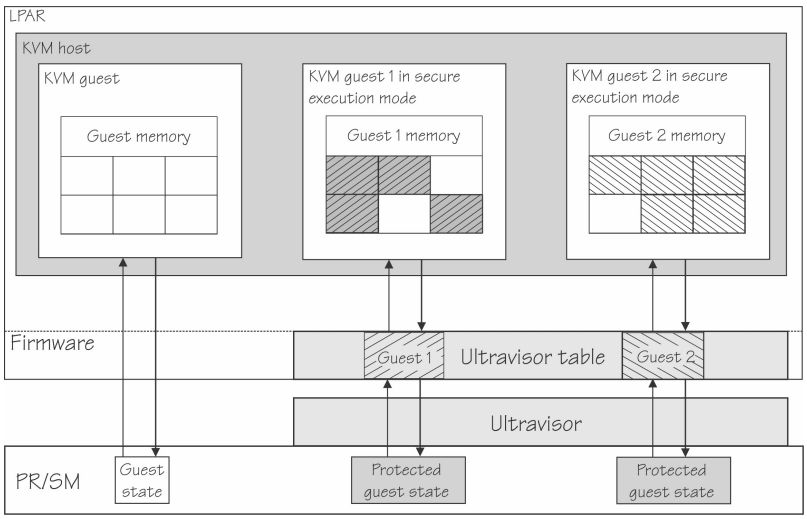


Figure 2. IBM Secure Execution protects guest memory and state

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 on page 7

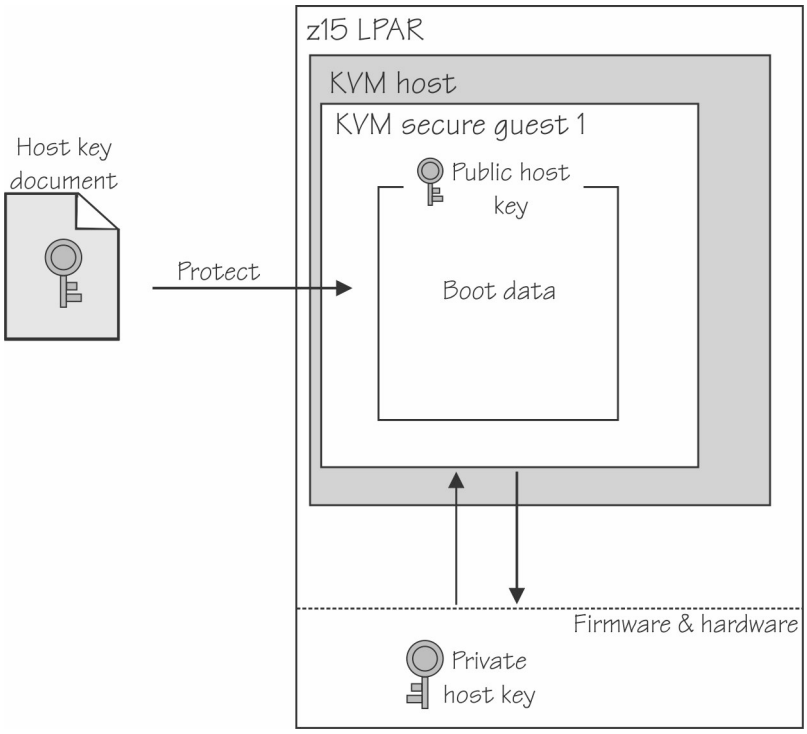


Figure 3. Encrypt files that are necessary for booting by using the host key document

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 Figure 5 on page 8, 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 on page 8.

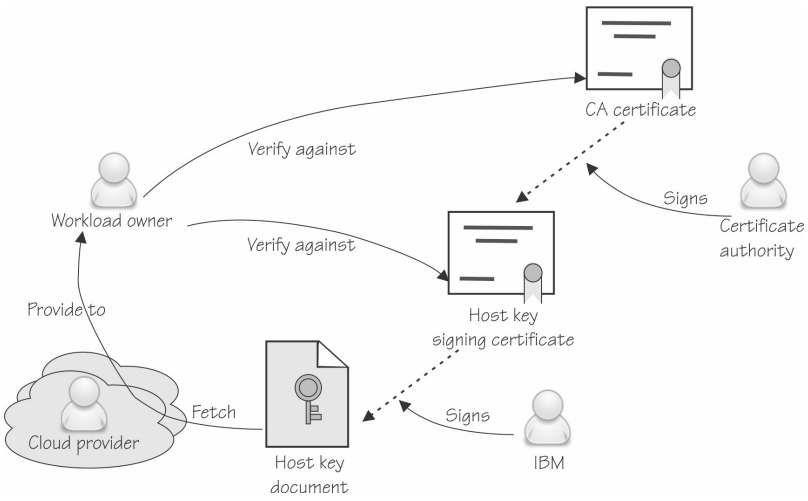


Figure 4. Chain of trust and key transfer using the host key document

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.

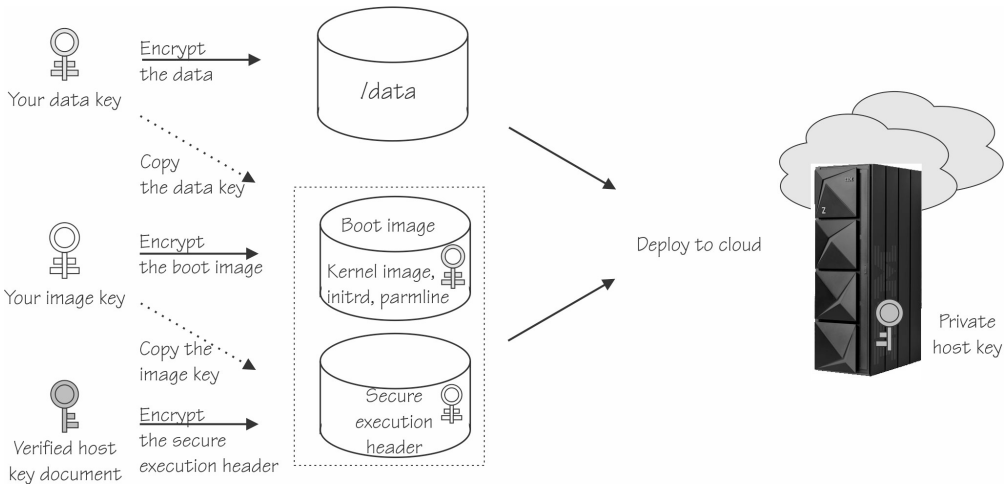


Figure 5. Conceptual key cascade for IBM Secure Execution

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 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 23
- 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.

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 Linux 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.

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.
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.

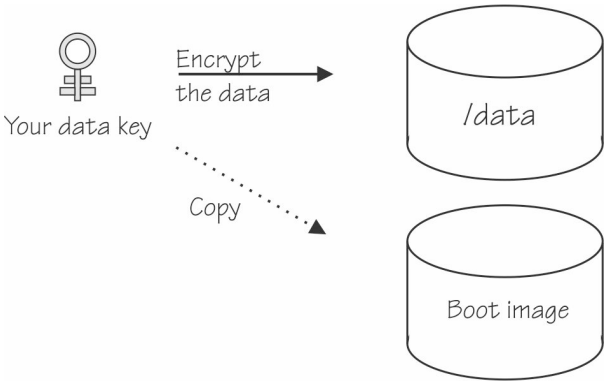


Figure 6. Data volumes for a workload need to be encrypted, using, for example, pervasive encryption

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.
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 bounce buffers 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 http://cdimage.ubuntu.com/releases/focal/release/ubuntu-20.04-live-server-s390x.iso 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-20.04-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.

   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
```

5. Your guest runs in the context of a virtual server. The virtual server defines the virtual hardware. Create a domain configuration-XML for the virtual server that defines the virtual hardware for your guest.

   In the domain configuration-XML, specify a name for the virtual server, and set `iommu="on"` to allow the use of bounce buffers on every element that represents a virtio device, for example, the `<disk>`, `<serial>`, and `<interface>` elements. Without the `iommu="on"` specification, the guest cannot work properly and will most likely crash.

   Optionally, you can boot by default from the bootable disk image by specifying `boot order=1`.

   For example, the following domain configuration-XML, called `secguest1.xml`, configures a virtual server called secguest1 with a guest that is bootable from a disk image that is called `secguest1.img` and allows bounce buffers:

```
<domain type="kvm">
   <name>secguest1</name>
   ...
   <devices>
      <disk type='file' device='disk'>
        <driver name='qemu' type='qcow2' cache="none" io="native" iommu='on'/>
        <source file='/var/lib/libvirt/images/focal.qcow2'/>
        <target dev='vda' bus='virtio'/>
        <boot order='1'/>
        <alias name='virtio-disk0'/>
      </disk>
      <interface type='network'>
        <mac address='52:54:00:36:d3:88'/>
        <source network='default' bridge='virbr0'/>
        <model type='virtio'/>
        <driver name='qemu' iommu='on'/>
      </interface>
      ...
   </devices>
```

   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
Make the guest secure with the **genprotimg** command as described in "Securing the guest" on page 16.

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 27.
- 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 14, 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. 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
   ```

   For more details about the **genprotimg** command, see Appendix A, "genprotimg - Generate an IBM Secure Execution image," on page 23.
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>
   ```

   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
   ```
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 25.

   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.

5. Securely remove the original kernel, RAM file system, and kernel parameter file as well as the related entries in the boot configuration using, for example, the **`shred`** command. Then re-run the boot configuration update. These files must not be revealed to the cloud provider, as they could potentially be used by an attacker to obtain secrets.
6. Send the prepared boot- and data images, and the domain configuration-XML to the cloud provider for deployment.

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.

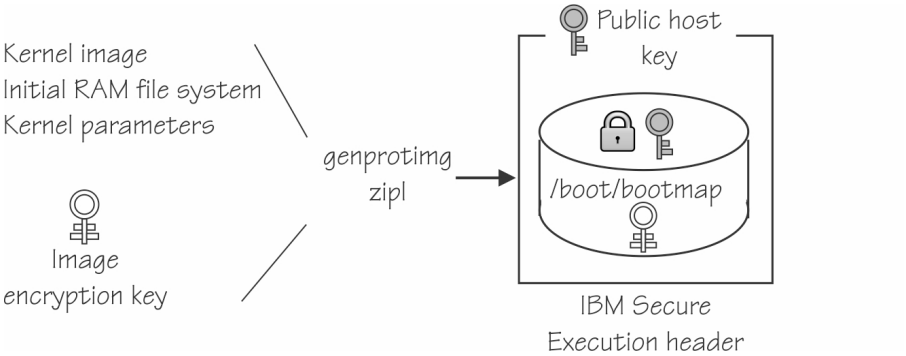


Figure 7. Boot data is consolidated and encrypted by IBM Secure Execution

What to do next

Supply your setup details to your provider, see “Communicating your setup to the provider” on page 17. 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

Test your image

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

Virtual server does not start: If your virtual server does not start, it might not have enough memory to unlock volumes during the boot process. 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.

The latter can also help to reduce the amount of memory that needs to be reserved for kdump.

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.

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. Under **Secure Execution for Linux**, 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.

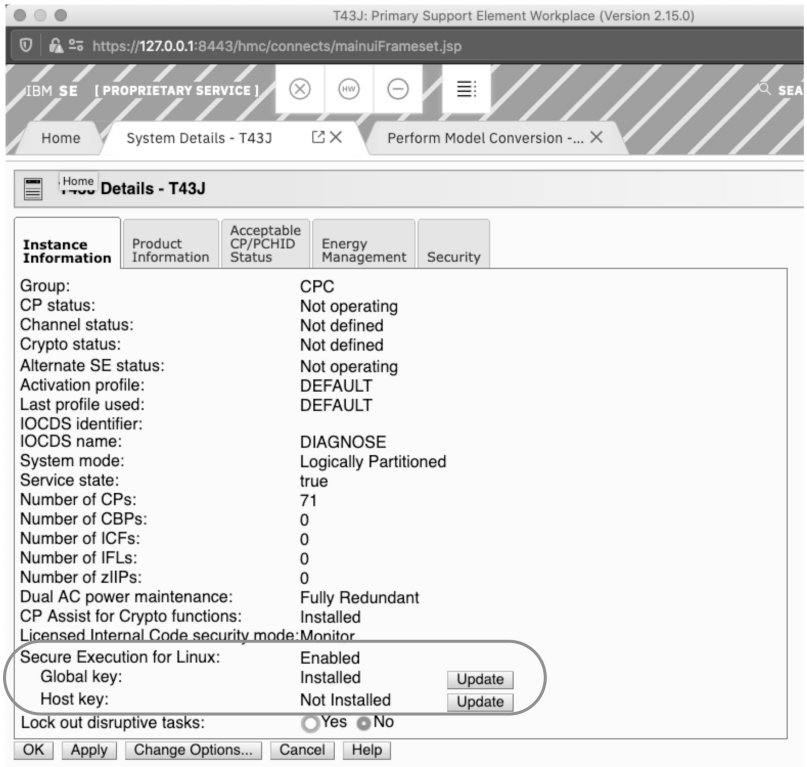


Figure 8. HMC showing how to import a key bundle

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 on page 21.

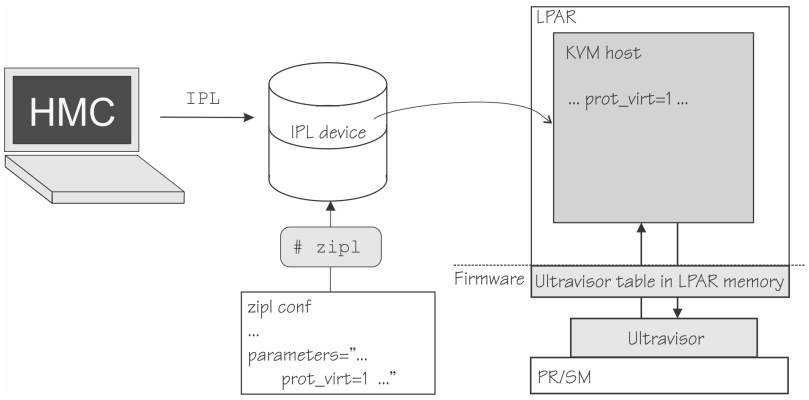


Figure 9. A KVM host is set up to run in IBM Secure Execution mode

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.

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.

   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.

   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. 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
```

4. 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.

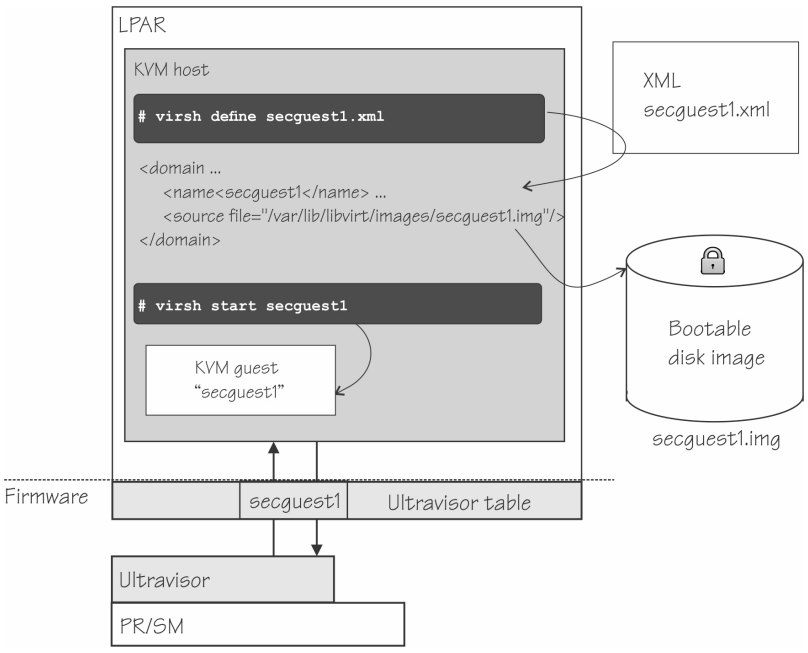
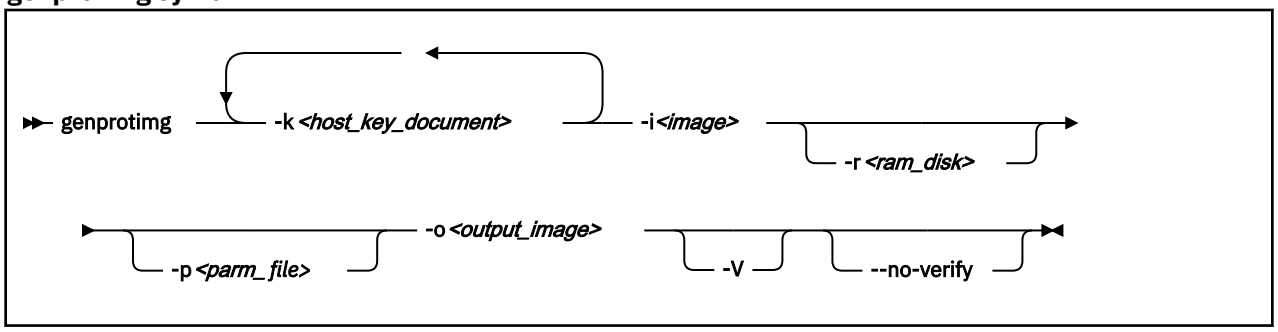


Figure 10. A KVM guest is created from a bootable image to run in IBM Secure Execution mode

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.

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.

-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
initrd /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 27.

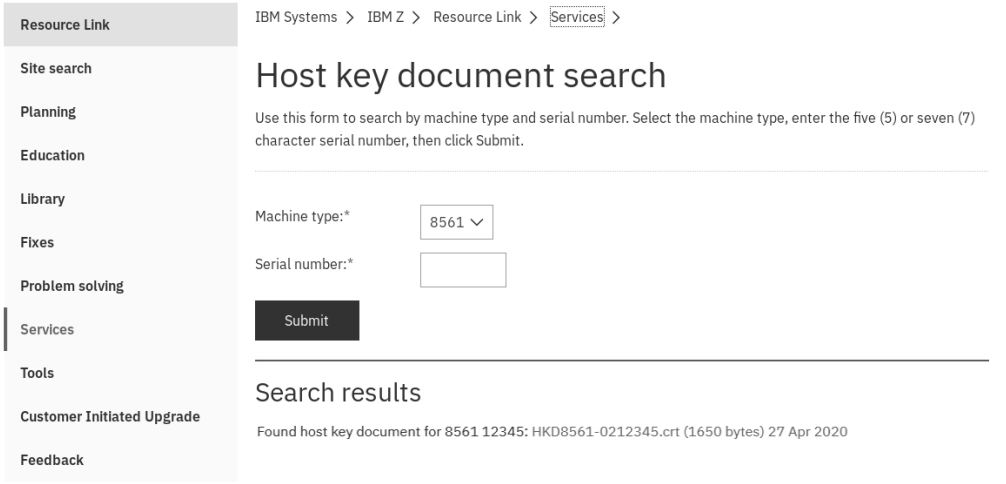


Figure 11. Host key document search page on Resource Link

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

Special Characters

/etc/securetty 12

B

bibliography v
boot configuration 25
boot record
    encrypted 23
booting
    components 12

C

cloud
    secure workload 11
cloud provider
    what needs to be communicated 17
commands
    genprotimg 23
configuration
    boot 25
console
    preventing login 12

D

disk
    DASD 12
    FCP-attached 12
    QCOW2 12
domain configuration-XML
    for cloud provider 17

G

genprotimg 23
guest preparation 12

H

host key document
    verifying 14
host, KVM v

I

IBM Secure Execution
    generate image 23
image
    generate IBM Secure Execution 23
initial RAM disk
    boot component 12

K

kernel
    boot component 12
key
    public host 12
key derivation method 17
KVM
    host v
    virtual server v
KVM guest
    prepare for secure execution 12
KVM host
    guest setup 21
    setup 20

L

login
    preventing 12
LUKS2
    key derivation method 17

M

memory
    kdump 17
    key derivation method 17

O

OpenSSH 12

P

parameters
    boot component 12
public host key 12

Q

QCOW2 12

S

secure execution
    boot configuration 25
setup
    KVM host 20

T

terminology v
tools
    genprotimg 23

V

verify
    host key document 14
virtual server, KVM v

W

workload, secure in the cloud 11

SC34-7721-00