Linux on IBM Z and IBM LinuxONE

Secure Boot
November 2023

SC34-7755-00

Note

Before using this information and the product it supports, read the information in “Notices” on page 25.

This edition applies to secure boot on Linux® on IBM Z® and IBM® LinuxONE in November 2023 , and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this publication

Learn about secure boot, how you can benefit from it, how to set up a Linux instance to use secure boot, how to manage certificates, and how to use your own signing keys.

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

```
https://www.ibm.com/docs/en/linux-on-systems?topic=linuxonibm/com.ibm.linux.z.lxsb/
lxsb_c_main.html
```

Distribution independence

The information in this publication is Linux distribution independent.

Where to get more information

Find more information about Linux on IBM Z and IBM LinuxONE.

For more information about booting and IPL, see *Device Drivers, Features, and Commands*. You can obtain the latest version of this publication on IBM Documentation at ibm.com/docs/en/linux-on-systems?topic=overview-device-drivers-features-commands.

For information about z/VM®, see the documentation for your z/VM version on IBM Documentation at ibm.com/docs/en/zvm or see the z/VM PDF library at www.ibm.com/vm/library/index.html.

NIAP GPOS certification and secure boot

The NIAP GPOS certification lists a number of requirements that an operating system must fulfill. One of these NIAP GPOS requirements named "Boot integrity" is fulfilled by secure boot for Linux on IBM Z.

Learn more about the NIAP General Purpose Operating System (GPOS) protection profile with these resources:

- What is NIAP/CCEVS? https://www.niap-ccevs.org/Ref/What_is_NIAP.CCEVS.cfm
- Protection Profile for General Purpose Operating Systems https://www.niap-ccevs.org/MMO/PP/-424-/

Other publications for Linux on IBM Z and IBM LinuxONE

You can find publications for Linux on IBM Z and IBM LinuxONE on IBM Documentation.

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
- *OpenSSL support for Linux on IBM Z and LinuxONE*, SC34-7732
- *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

Conventions used in this publication

A summary of the styles, highlighting, and assumptions used throughout this publication.

Authority

Most of the tasks described in this document require a user with root authority. In particular, writing to `procfs`, and writing to most of the described sysfs attributes requires root authority.

Throughout this document, it is assumed that you have root authority. Command examples are illustrated as follows:

- A *#* command prompt indicates that root authority is required:

```
# ...
```

- A $ command prompt indicates that root authority is not needed:

```
$ ...
```

Terminology

In this publication, the term *booting* is used for running boot loader code that loads the Linux operating system. *IPL* is used for issuing an IPL command to load boot loader code or a stand-alone dump utility.

sysfs and procfs

In this publication, the mount point for the virtual Linux file system `sysfs` is assumed to be `/sys`. Correspondingly, the mount point for `procfs` is assumed to be `/proc`.

Highlighting

This publication uses the following highlighting styles:

- Paths and URLs are highlighted in `monospace`.
- Variables are highlighted in *<italics within angled brackets>*.
- Commands in text are highlighted in **`monospace bold`**.
- Input and output as normally seen on a computer screen is shown

```
within a screen frame.
Prompts are shown as hash signs:
#
```

Chapter 1. Introduction to secure boot on IBM Z and IBM LinuxONE

Use secure boot to ensure the code that you boot is trustworthy.

Secure boot is a security standard that ensures that a Linux instance boots using only trusted software. Software is considered trusted if it is signed. When the instance starts, the signatures of boot components that contain code are checked. With secure boot enabled, booting fails if a component that contains code is not signed or cannot be verified.

Secure boot is a platform capability that verifies that code that is loaded during boot:

- Originates from a trusted source
- Was not modified.

An operating system whose code cannot be verified fails to boot when booted using secure boot.

For Linux on IBM Z, the trusted source of the boot code is typically a Linux distributor, such as Red Hat®, SUSE, or Canonical. However, you can also sign boot files of a custom Linux system, including a custom kernel, yourself for use with secure boot.

Signed code

Distributors sign their Linux kernel and boot loader binary files with a private key and ship the resulting code signature as part of the corresponding software package. Distributors also provide the corresponding public keys included in X.509 v3 certificates. These public keys are used by the platform bootloader firmware to verify the code signatures during boot.

NIAP GPOS certification

The secure boot capability is a requirement for a certification of Linux on IBM Z according to the National Information Assurance Partnership (NIAP) General Purpose Operating System (GPOS) protection profile. This certification is a mandatory prerequisite for deploying an operating system in certain industries and in particular for certain US specific government offices.

For more information about NIAP GPOS, see the references in “NIAP GPOS certification and secure boot” on page v.

Who should read this information and secure boot roles

Most of the information in this document is intended for Linux administrators who want to configure Linux on IBM Z. Some information might also be of interest for IT architects, who need to understand the secure-boot related requirements for running Linux on IBM Z in an environment that requires a NIAP GPOS certification.

Several persons in an organization work with secure boot, the certificates, and signatures involved:

- The person who signs Linux boot files and kernel modules; this is typically a Linux administrator. The signing person has access to the private key for which the public key is uploaded to the HMC.
- The person who uploads the public-key certificate to the HMC; this can be a hardware administrator.

Supported environments

Secure boot requires hardware and software support.

Support for secure boot of Linux on IBM Z was introduced with IBM z15™ for boot of Linux in LPAR mode from SCSI devices, using certificates from Red Hat, SUSE, and Canonical that were integrated in IBM Z firmware.

Since then, secure boot was continuously extended, and now includes support of:

- Locally-attached NVMe and ECKD DASD boot devices
- Linux running as a z/VM guest (for SCSI and ECKD DASD boot devices)
- Custom certificates that are provided to the hardware operator, and that the operator can then upload and assign to LPARs.

Table 1 on page 2 lists the supported boot media by hardware and Linux distribution.

Table 1. Secure boot hardware and software requirements

Boot media	Minimum hardware or hypervisor	Minimum Linux version
FCP-attached SCSI disk	IBM z15 IBM LinuxONE III	Red Hat Enterprise Linux 8.1 SUSE Linux Enterprise Server 15 SP2 Ubuntu 19.10
NVMe	IBM LinuxONE III	Red Hat Enterprise Linux 8.3 SUSE Linux Enterprise Server 15 SP3 Ubuntu 20.10
ECKD DASD	IBM z16 IBM® LinuxONE 4	Red Hat Enterprise Linux 8.8, and 9.2 SUSE Linux Enterprise Server 15 SP5 Ubuntu 23.04
SCSI and ECKD DASD for z/VM guests	IBM z16 IBM® LinuxONE 4 z/VM 7.3 with PTFs	Red Hat Enterprise Linux 8.8, and 9.2 SUSE Linux Enterprise Server 15 SP5 Ubuntu 23.04

Environments that support secure boot still allow a standard (non-secure) boot of Linux. Choose a secure boot IPL option to boot an operating system with the secure boot function.

If this option is chosen, an unsigned operating system, or an operating system whose signature cannot be verified, fails to boot. If you do not choose the secure boot IPL option, booting a signed operating system whose signature cannot be verified triggers a warning that is logged on the HMC.

Secure boot restrictions

Restrictions apply when running Linux with secure boot enabled. Once started in secure boot mode, a Linux kernel runs in *lockdown* mode to improve security.

Signed kernel modules

In a kernel prepared for secure boot, all kernel modules must be signed either:

- By the distributor.
- By the person who signs modules and code, who has access to the private key for which the associated public key was uploaded to the HMC and assigned to the LPAR.

You cannot load modules that are not signed.

Restricted kernel interfaces

Access to certain kernel interfaces is restricted, with impact on for example the **hyptop** command.

Verifying Linux kernel lockdown mode

To check whether Linux is running in lockdown mode, check the sysfs attribute `lockdown`:

```
$ cat /sys/kernel/security/lockdown
none [integrity]
```

The above result indicates that lockdown is in effect. The following example indicates that lockdown is off:

```
$ cat /sys/kernel/security/lockdown
[none] integrity
```

Alternatively, search the Linux kernel boot messages for a message containing the text `kernel_lockdown`:

```
$ dmesg -t | grep kernel_lockdown
Kernel is locked down from Secure IPL mode; see man kernel_lockdown.7
```

Chapter 2. Using secure boot on Linux on IBM Z and IBM LinuxONE

Using secure boot includes finding out if your environment supports it, preparing a boot device, initiating a boot procedure, and verifying that Linux was booted using secure boot.

Finding out if your environment supports secure boot

Check the `has_secure` sysfs attribute to determine if your environment supports secure boot.

Before you begin

Linux must be booted from a supported disk type, see Table 1 on page 2.

Procedure

Read the `has_secure` sysfs attribute.

For example:

```
# cat /sys/firmware/ipl/has_secure
1
```

Results

A value of "1" indicates that the environment supports secure boot, otherwise the result is "0".

If you boot from a supported disk type, but the attribute does not exist, the version of Linux might be too old.

Preparing a boot device

Use the **zipl** command to prepare a boot device for secure boot.

About this task

The **zipl** boot loader program prepares Linux disks for boot. The **zipl** command automatically installs the secure boot disk format when you run it in an environment that supports secure boot, and if code signatures are available.

Tip: You can instruct **zipl** to always use the secure boot format by specifying the `--secure=1` command-line parameter.

A Linux SCSI or NVMe boot disk prepared for secure boot does not boot in an environment that does not support secure boot.

The following commands can be used to reinstall the boot loader while forcing the use of secure boot. To reinstall without secure boot support, change the value "1" to "0".

Red Hat Enterprise Linux and Ubuntu

On Red Hat Enterprise Linux and Ubuntu, use the **zipl** command with the `--secure=1` option to prepare a boot device.

Procedure

Issue the following command:

```
# zipl --secure=1
```

SUSE Linux Enterprise Server

On SUSE Linux Enterprise Server, use GRUB 2 to prepare a boot device.

Procedure

1. Add the following line to file `/etc/default/grub`.

   ```
   SUSE_SECURE_BOOT=1
   ```

2. Re-run the boot loader installation.

   ```
   # grub2-install
   ```

Initiating secure boot

You can boot securely from a supported boot device.

Initiating a secure boot from an LPAR

Initiate a boot from LPAR from the HMC or SE.

Procedure

1. Open the HMC Load panel for the target LPAR and select the following settings:
   For SCSI or NVME devices:
   - Device Type: SCSI or NVMe
   - Check the **Enable Secure Boot** checkbox.

   For ECKD DASD devices:
   - Device type: ECKD
   - IPL type: List-directed
   - Check the **Enable Secure Boot** checkbox.

   The HMC load panel for loading an ECKD DASD is shown in Figure 1 on page 7

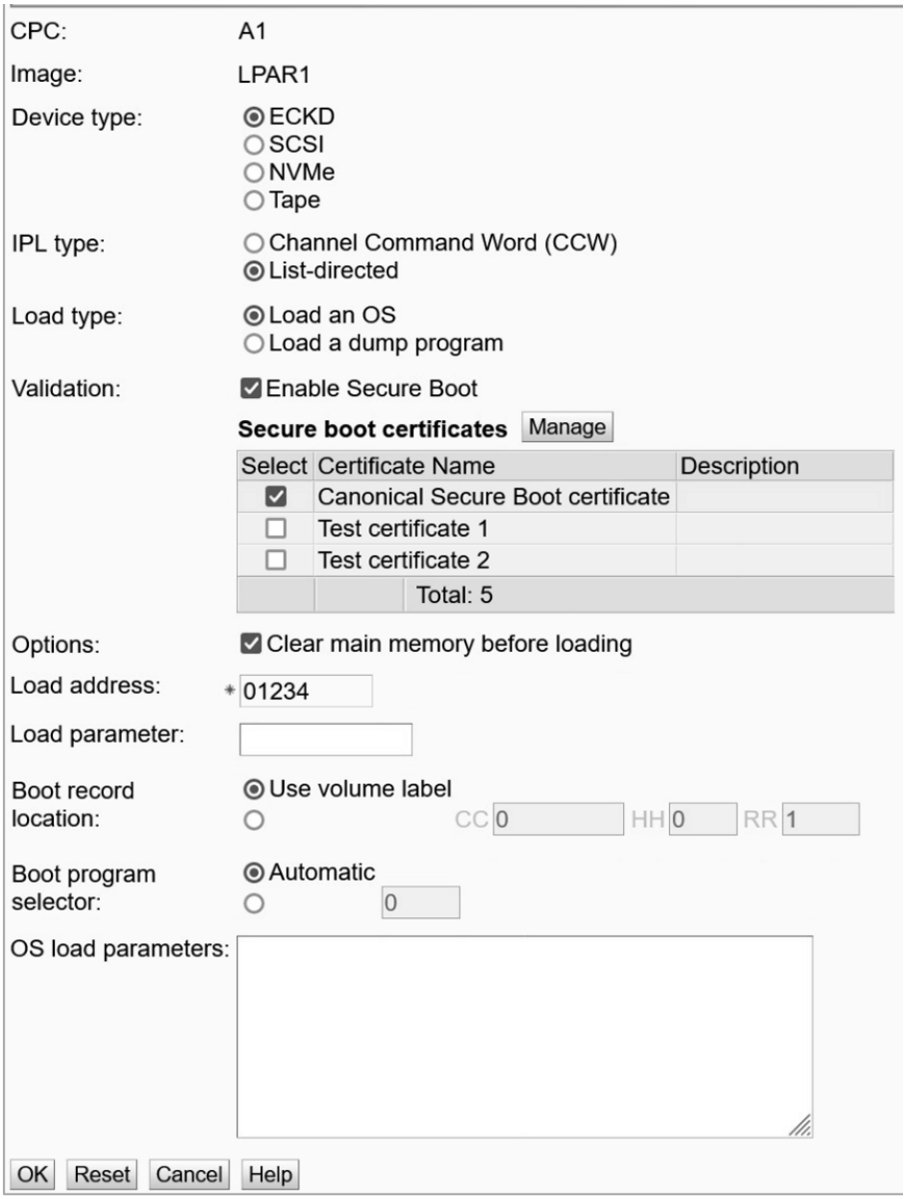

Figure 1. HMC load panel for LPAR boot

2. To initiate secure boot, click **OK**.

Initiating secure boot from z/VM

z/VM guest secure boot is supported for SCSI and ECKD DASD devices.

Before you begin

For secure boot of z/VM guests, you need the following:

- IBM z16 or IBM® LinuxONE 4.
- z/VM 7.3 with the PTF for APARs VM66434 (CP), VM66424 (DirMaint), and VM66650 (SMAPI), see https://www.ibm.com/docs/en/zvm/7.3?topic=apars-apar-guest-secure-ipl.
- One of these distributions, or later:
  - Red Hat Enterprise Linux 8.8, and 9.2
  - SUSE Linux Enterprise Server 15 SP5
  - Ubuntu 23.04

Procedure

- In CP, enter the following commands to boot from a SCSI device.
  For example:

```
SET LOADDEV SCSI DEVICE 7412 SECURE
SET LOADDEV PORT 50050763 00c20b8e LUN 52410000 00000000
IPL LOADDEV
```

  where the SCSI device is defined by:
  - Device number of the FCP device that is used, for example, `0.0.7412`
  - FCP port number `50050763 00c20b8e`
  - FCP logical unit number (LUN) `52410000 00000000`
- To boot from an ECKD DASD device, enter the following commands.
  For example:

```
SET LOADDEV ECKD DEVICE 7e78 SECURE
IPL LOADDEV
```

  where 7e78 denotes the device number of the DASD.

Verifying that Linux was booted securely

Use the `secure` sysfs attribute to determine whether Linux was booted securely.

About this task

A reliable method of checking whether a Linux instance was booted using secure boot is to use the HMC **Load** panel or the z/VM **CP LOADDEV** command. However, to quickly check on a running Linux instance, you can use this procedure.

Procedure

On your Linux instance, issue the following command to check whether the instance was booted using Secure Boot:

```
# cat /sys/firmware/ipl/secure
1
```

A value of "1" indicates that Linux was booted using secure boot, otherwise the result is "0".

Alternatively, search the Linux kernel console log for a message containing the text "Secure-IPL enabled":

```
# dmesg -t | grep Secure-IPL
setup: Linux is running with Secure-IPL enabled
```

Note: These indications from within the Linux instance should be used for informational purposes only, as they could be forged if the system is compromised.

Chapter 3. Managing secure boot certificates

Manage certificates for an LPAR, learn about Linux keyrings, how to enable third-party certificates, how to obtain distributor certificates, and know which certificates are built into the firmware.

Managing a secure boot certificate for an LPAR

Use the HMC **Secure Boot Certificate Management** task to upload custom secure boot certificates.

Before you begin

A secure boot certificate must meet the following requirements before it can be uploaded:

- It must be an X.509 v3 certificate.
- It must be PEM or DER encoded.
- It must have a maximum file size of 20 kB.
- Its file name extension must be `.pem`, `.der`, `.cer` or `.crt`.
- The certificate must not be expired.
- The certificate can be signed by a Certificate Authority (CA), or self-signed.

Note: No CA certificate validation is performed during import.

Additional restrictions regarding the use of digital signature algorithms may be imposed by the actual operating system version that is booted.

About this task

As of IBM z16 and IBM® LinuxONE 4 with the update from May 2023 you can upload certificates. You can then assign these certificates to an LPAR for use with secure boot.

For operators, advantages of this feature include the ability:

- to react to secure boot certificate revocation according to the policies of their company (that is, without being forced to wait for a new firmware update).
- to restrict the securely bootable kernels to specific Linux distributors.
- to configure the system to securely boot Linux with kernels signed by the customer or a trusted vendor of the customer's choice.

Procedure

1. On the HMC, click **Search**.

   For detailed steps without using search, see "Detailed steps on the HMC" on page 10 or "Detailed steps in DPM mode" on page 11.
2. In the search field, enter "Secure Boot".
3. From the search results list, select **Secure boot certificate management**.
4. Click **Import**.
5. Click **File Upload**, and **Choose file** on the panel shown in Figure 2 on page 10, then click **Next**.

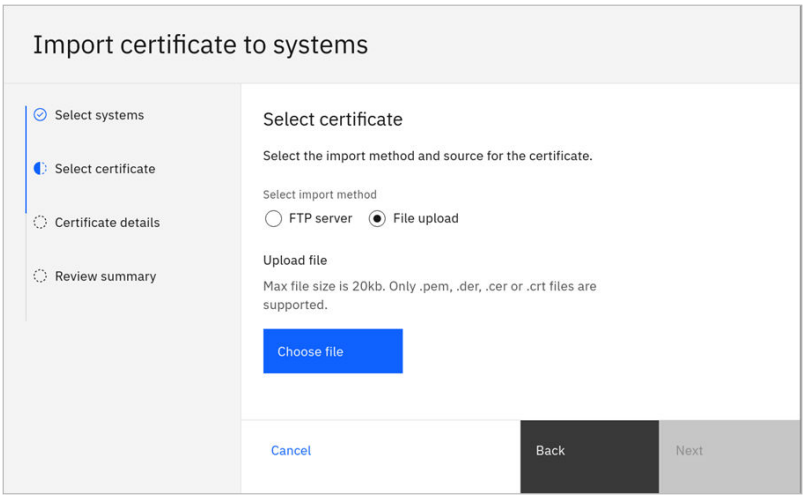

Figure 2. HMC panel for importing Secure Boot certificates

6. On the next panel, enter a name and optional description for the certificate, then click **Next**.
7. Review the entered data and finalize the import by clicking **Import certificate**.

Detailed steps on the HMC

Find the HMC **Secure Boot Certificate Management** task on an HMC.

Procedure

1. On the HMC, select the LPAR you want to work with.
2. Choose the "Load" task.
3. On the Load panel,select **Device type**. An example of the Load panel is shown in Figure 3 on page 10
4. For ECKD devices, select **IPL type** List-directed. The **Enable Secure Boot** check box appears.
5. Check the **Enable Secure Boot** checkbox. The **Secure boot certificates** UI control appears.
6. Click **Manage**.

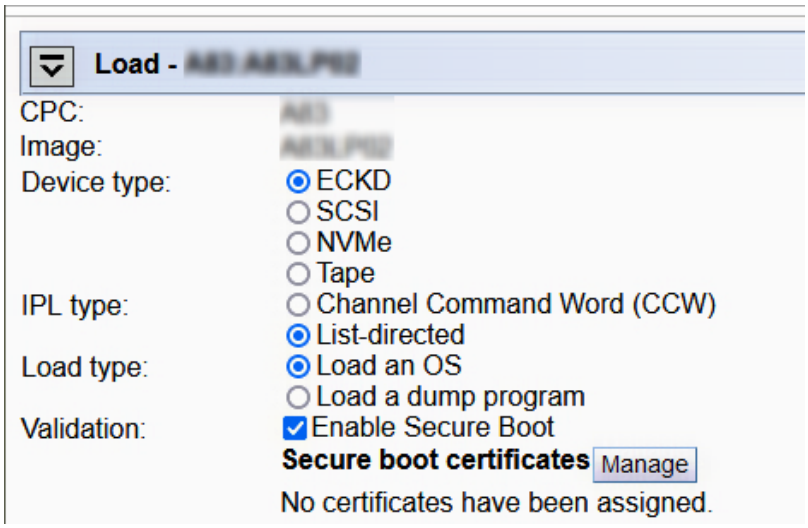

Figure 3. HMC load panel with Enable Secure Boot UI control

What to do next

Continue with “4” on page 9 in “Managing a secure boot certificate for an LPAR ” on page 9.

Detailed steps in DPM mode

Find the HMC **Secure Boot Certificate Management** task when working in DPM mode.

Procedure

1. Select the partition you want to work with.
2. Choose the "Partition details" task.
3. Go to the **Boot** section.
4. Find the **Certificates** UI control.
5. Click **Manage**.

What to do next

Continue with “4” on page 9 in “Managing a secure boot certificate for an LPAR ” on page 9.

Secure boot certificates and Linux keyrings

The Linux kernel maintains lists of encryption and authentication keys in so-called *kernel keyrings*.

One such keyring is the *platform keyring*. On the Linux distributions listed in “Supported environments” on page 1, Linux automatically adds all secure boot certificates used by IBM Z firmware during secure boot to the platform keyring.

To use the **`keyctl`** command, you require the keyutils package, available from: https://git.kernel.org/pub/scm/linux/kernel/git/dhowells/keyutils.git

Note: Secure boot certificates are not added to the platform keyring in the upstream kernel.

Use the following command to display the list of keys in the platform keyring:

```
# keyctl show %:.platform
Keyring
 104926345 ---lswrv      0     0  keyring: .platform
 629804537 ---lswrv      0     0   \_ asymmetric: SUSE Linux Enterprise Secure Boot Signkey:
a746b64b6cb71f13385638055f46162bac632acd
 241500789 ---lswrv      0     0   \_ asymmetric: Canonical Ltd. Secure Boot Signing (ZIPL,
2019): 54d6f4e263d8c44592aca76962cf13795f0c30da
 889219187 ---lswrv      0     0   \_ asymmetric: Red Hat Secure Boot (signing key 2):
80d598d7d868efaaefb968534a99b3d7490da9e6
  90870925 ---lswrv      0     0   \_ asymmetric: Red Hat Secure Boot Signing 3 (beta):
65c5becae6596afd6c71c4a798c6258d7b6705d0
```

Enabling third-party modules in a secure boot environment

You can enable third-party kernel modules by uploading their public-key certificates.

About this task

When Linux is booted in secure boot mode, kernel-module signature-verification is automatically enforced. To perform the verification of module signatures, current **Linux** distributions only use public keys in the kernel platform and built-in keyrings.

This prevents loading of kernel modules provided by third-party vendors, even if these modules are correctly signed, if the public key of the vendor is not contained in the kernel platform key ring.

Procedure

1. To enable loading of signed third-party kernel modules, upload the corresponding third-party public key certificate to the IBM Z HMC.

   Use the **Secure Boot Certificate management task**, similar to how Linux distributor certificates are uploaded.

2. Reboot Linux.

Results

During boot, the third-party key is added to the Linux platform keyring, from where it is used to verify kernel module signatures.

Obtaining Linux distributor secure boot certificates

You can fetch certificates from Linux distributors.

About this task

Should you need to configure an LPARs with certificates of your choice, for example, using a new certificate after a Linux distributor has revoked their previous certificate, you must fetch the corresponding certificates from the distributor:

- Red Hat, "Red Hat Secure Boot (signing key 2)", at https://access.redhat.com/security/team/key
- SUSE, "SUSE Secure Boot Signing Key", at https://www.suse.com/support/security/keys
- Canonical, file "sipl.x509", at http://us.ports.ubuntu.com/ubuntu-ports/dists/jammy/main/signed/linux-s390x/current/signed.tar.gz. Replace "jammy" with the first part of the Ubuntu distribution code name. For example for Ubuntu 22.04 = jammy, because the code name is "Jammy Jellyfish".

Procedure

1. Verify the validity of the certificate.
   You can use openSSL tools to do this.
2. Upload the certificate to the HMC.
3. Assign it to the target LPAR.

Certificates built into the firmware

When no custom certificate is assigned to an LPAR during secure boot, IBM Z uses the Linux distributor certificates that are integrated in IBM Z firmware.

Deprecated function:

The use of firmware-integrated certificates is deprecated, and will be removed in the future. In particular, in the future a firmware-integrated certificate might be revoked by the distributor, and therefore no longer be valid for verifications.

For information on how to obtain a valid key from a distributor, see “Obtaining Linux distributor secure boot certificates” on page 12.

Chapter 4. Signing boot files and modules with your own key

You can sign Linux boot files and kernel modules with your private keys.

Before you begin

Ensure that the `openssl` package is installed on your system.

About this task

Signing with your own key can be useful in certain scenarios, such as:

- Strict control of what Linux OS level is allowed to run in an LPAR or z/VM guest.
- Secure boot of custom-built Linux kernels, modules, and boot loaders.

Remember: Using your own signatures adds a constant maintenance effort because signatures need to be refreshed whenever a binary file that needs to be signed is updated. This includes the Linux kernel and s390-tools updates.

The following sections demonstrate the basic procedure to manually sign Linux boot files and kernel modules. Additional aspects and security best practices outside the scope of this document must be followed for the resulting signatures to be considered secure.

At a minimum, ensure that all security updates are applied on the system used for signing, and that the private key is protected from unauthorized access, for example through encryption, or by storing it in hardware-protected key-management systems.

Creating a private key

A private key is the base for creating digital signatures.

About this task

There are different cryptographic algorithms and key parameters that influence the format of the private key and resulting signatures. Take care to choose key parameters that are compatible with both IBM Z secure boot firmware and Linux kernel module verification.

The common key format used for Linux digital signatures is RSA with a key length of 2048 bits. This format is also supported by IBM Z firmware.

Procedure

Use the **openssl genpkey** command to create a private key.

For example, issue:

```
$ openssl genpkey -algorithm RSA \
                  -pkeyopt rsa_keygen_bits:2048 \
                  -out private-key.pem
```

The above example generates a new private key using the RSA algorithm with key length of 2048 bits, and stores the result in a file named `private-key.pem`.

What to do next

You can now use the private key for a self-signed certificate, see “Creating a self-signed certificate” on page 14.

Creating a self-signed certificate

Certificates can be either self-signed, or signed by a certificate authority (CA).

About this task

A public-key certificate contains the public key that is required to validate signatures as well as information that identifies the subject that owns the private key. For the purposes of Linux secure boot and module signing, a self-signed certificate is sufficient and is used in the examples that are shown here.

Procedure

1. Create a text file named `cert-params.conf` with the following content:

```
[req]
distinguished_name   = subject
x509_extensions      = x509_ext
prompt               = no

[subject]
commonName           = "<name>"
emailAddress         = "<e-mail>"

[x509_ext]
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid,issuer
basicConstraints        = critical, CA:false
keyUsage                = digitalSignature
extendedKeyUsage        = codeSigning
```

   This file defines certificate parameters and identity information for the new certificate.
2. Replace the *<name>* and *<e-mail>* placeholders with actual identity information of the individual or organization that owns the private key.
3. Create the certificate.

   You need the identity information that is specified in the configuration file `cert-params.conf` and the private key `private-key.pem`. A public key is derived from the private key and written to `cert.pem`. Issue a command like the following example:

```
$ openssl req -new -x509 -sha256 -days 365 \
              -config cert-params.conf \
              -key private-key.pem \
              -out cert.pem
```

   For security reasons, each certificate is assigned an expiration date after which it is no longer considered valid. The certificate that is created in this example is set to expire in one year (365 days) from the date of creation.

Results

You can display the contents of the resulting certificate in a textual format by using the following command:

```
$ openssl x509 -in cert.pem -text -noout
```

What to do next

To activate this certificate for use with Linux secure boot and kernel-module verification, upload it to the HMC Secure Boot certificate store as outlined in “Managing a secure boot certificate for an LPAR ” on page 9.

Linux files that require a signature

Certain files must have a signature to work with secure boot.

If a required file lacks a signature and secure boot was selected, the IPL fails. For secure boot, the following Linux files must be signed:

Table 2. Linux files that require a signature

Distribution	File names	Package
Red Hat Enterprise Linux	• `/boot/vmlinuz-<version>` • `/lib/s390-tools/stage3.bin`	• kernel-core • s390utils-core
SUSE Linux Enterprise Server	• `/boot/image-<version>` • `/boot/zipl/image-<version>` • `/lib/s390-tools/stage3.bin`	• kernel-default • kernel-default (updated by grub2-install) • s390-tools
Ubuntu Server	• `/boot/vmlinuz-<version>` • `/lib/s390-tools/stage3.bin`	• linux-image • s390-tools-signed

Adding a signature

Use the **`sign-file`** command to add a code signature to Linux boot files.

Before you begin

Depending on your Linux distribution, the **`sign-file`** command might be packaged as part of the `kernel-devel` package. If the command is not available, you must compile it from source code.

For source-code download and compilation of the **`sign-file`** command, the following packages are required:

- gcc
- openssl-devel
- curl

About this task

If a new signature is added to an already signed file, only the new signature is used for signature validation, but the old signature data remains between the original file data and the new signature.

Old signature data does not typically cause a problem during boot, but the growing file size might lead to an overlap at boot time, causing boot failures.

Tip: Remove any existing code signature before adding a new signature. See also “Removing a code signature” on page 17.

Procedure

1. If your distribution does not contain the **`sign-file`** command, download the source code and compile the command.

   To obtain the latest **`sign-file`** source code from the upstream Linux kernel source repository, issue the following command:

```
# curl -O https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/plain/scripts/
sign-file.c
```

2. To compile the command, issue:

```
# gcc sign-file.c -o sign-file -lcrypto
```

3. Sign a boot file with the **sign-file** command.
   Issue a command of the following form:

```
# sign-file sha256 <private_key_file.pem> <public_key_cert.pem> <file>
```

   where:

   - *<private_key_file.pem>* is the file containing your private key.
   - *<public_key_cert.pem>* is the file containing your public key.
   - *<file>* is the boot or module file you want to sign.

   For example, issue:

```
# sign-file sha256 private-key.pem cert.pem /lib/s390-tools/stage3.bin
```

   In the example, the command signs a boot file using a private-key file `private-key.pem` and a public-key certificate `cert.pem`.

Results

The resulting data consists of a DER-encoded PKCS#7 signature and additional Linux-specific information. This data is appended to the end of the original file as a signature trailer, increasing the original file size.

Verifying a code signature

Look for a text marker to verify that a file is signed.

About this task

The signature data that is appended to a signed file consists of the following parts:

Table 3. Linux signature-trailer format

Offset from end of file	Description
-(*<L>*+40)	Signature data in DER-encoded PKCS#7 format
-40	8 bytes of Linux-kernel-specific metadata
-32	A 4-byte integer that specifies the length *<L>* of the PKCS#7 signature
-28	An ASCII text marker “~Module signature appended~”

Procedure

Display the last 28 bytes of a file to check whether it is signed.

If the output contains the signature text marker `~Module signature appended~` then the file is signed.

Issue a command of the following form:

```
# tail -c 28 <file>
~Module signature appended~
```

where *<file>* is the file that you want to check.

Extracting a code signature

You can extract the DER-encoded PKCS#7 signature data from a signed file.

About this task

You can use the data found in the signature trailer to extract the DER-encoded PKCS#7 signature data from a signed file.

Procedure

1. The following commands store the signature data:
   Define the signed file as a local variable:

```
$ FILE="<file>"
```

2. Calculate the length, L, of the signature:

```
$ L=$(( $(tail -c 32 "$FILE" | hexdump -n 4 -ve '"0x%x"') ))
```

3. Use the **tail** command to extract the signature, located at length + 40 bytes from the end of the file, and store it a file called `signature.der`:

```
$ tail -c $(( L+40 )) "$FILE" > signature.der
```

4. Optional: Display the extracted signature.
   Use the following **openssl** command to display the signature stored in `signature.der`:

```
$ openssl cms -cmsout -print -noout \
              -inform der -in signature.der
```

Removing a code signature

Remove a code signature by copying the unsigned part of a file to a new file.

Before you begin

Check that the file is actually signed before performing these steps, see "Verifying a code signature" on page 16. Otherwise, the original file contents will be corrupted.

Procedure

1. The following commands copy the unsigned part of a file to a new file:
   Define the file you want to remove the signature from as a local variable:

```
$ FILE="<file>"
```

2. Calculate the length, L, of the signature:

```
$ L=$(( $(tail -c 32 "$FILE" | hexdump -n 4 -ve '"0x%x"') ))
```

3. Calculate the size of the file, for example with the **stat** command:

```
$ F=$(stat -c%s "$FILE")
```

4. Write the beginning of the file up to the signature to a new file, here called `$FILE.orig`:

```
$ head -c $(( F-L-40 )) "$FILE" >"$FILE".orig
```

Manually verifying a code signature

You can manually verify whether file contents and code signature match.

Procedure

1. Extract the code signature.

   Follow the steps in “Extracting a code signature” on page 17.
2. Remove the code signature to obtain an unsigned file.

   Follow the steps in “Removing a code signature” on page 17.
3. Check that the signature and file data match.

   Use the extracted signature `signature.der`, the unsigned file `$FILE.orig`, the public key certificate `cert.pem`, and the following **openssl** command to check whether the signature and file data match:

```
$ openssl cms -verify -binary -nointern -noverify \
              -certfile cert.pem \
              -inform der -in signature.der \
              -content $FILE.orig \
              -out /dev/null
CMS Verification successful
```

Results

An output line of "CMS Verification successful" indicates that signature and file contents match.

Chapter 5. Troubleshooting secure boot

Problems that you can fix include expired certificates, obsolete packages, missing signatures, and unsupported environments.

Determining successful preparation

Make sure that the boot media was successfully prepared for secure boot.

Procedure

Use **`zipl`** with the `--verbose` option to determine whether a Linux boot volume was successfully prepared for secure boot:

```
$ sudo zipl --verbose --secure=1
...
Secure boot support: yes
...
Adding #1: IPL section 'ubuntu' (default)
  initial ramdisk...: /boot/initrd.img
  signature for.....: /lib/s390-tools/stage3.bin
  kernel image......: /boot/vmlinuz
  signature for.....: /boot/vmlinuz
...
Preparing boot device for CCW- and LD-IPL: dasda (1234).
...
```

Watch for the following output lines:

- `Secure boot support: yes`

  This line indicates that the environment supports secure boot.
- `signature for...: <filename>`

  This line indicates that the listed boot file contains a secure boot signature. For a successful boot, this message must appear twice; once for `stage3.bin` and once for the Linux kernel, typically named `vmlinuz` or `image`.
- `Preparing boot device for CCW- and LD-IPL`

  This message occurs only for DASD boot devices and indicates that the DASD was prepared both for traditional CCW boot, and for LD-IPL boot, which is required for secure boot.

Results

On a successful secure boot, the following lines are displayed on the HMC operating system message console, or the z/VM guest console, before any operating system output.

```
IPB received.
IPB sent.
System version 9.
Watchdog enabled.
Running 'ZBootLoader' version '3.1.5' level 'D51C.D51C_328.13'.
OK00000000 Success
```

For more information about messages on the HMC, see SC28-7046-00: *IPL Machine Loader Messages*, available at: https://www.ibm.com/support/pages/sites/default/files/2023-06/SC28-7046-00.pdf

Reinstalling Linux boot-file packages

You can update and reinstall boot-file packages.

About this task

If Linux boot files were corrupted, or if boot-file signatures are invalid, for example due to an expired secure boot certificate, proceed as follows:

1. Update the relevant packages to their latest versions.
2. If the package is already at the latest version, reinstall the package.

Procedure

- Red Hat Enterprise Linux

  To update packages to their latest versions, issue:

  ```
  # dnf update kernel-core s390utils-core
  ```

  To reinstall packages, issue:

  ```
  # dnf reinstall kernel-core s390utils-core
  ```

- SUSE Linux Enterprise Server

  To update packages to their latest version, issue:

  ```
  # zypper update kernel-default s390-tools
  ```

  To reinstall packages, issue:

  ```
  # zypper install -f kernel-default s390-tools
  # grub2-install
  ```

- Ubuntu

  To update packages to their latest versions, issue:

  ```
  # apt update
  # apt upgrade linux-image-generic s390-tools
  ```

  To reinstall packages, issue:

  ```
  # apt install --reinstall linux-image-$(uname -r) s390-tools-signed
  ```

Common problems and their solutions

Learn to solve some common problems.

Unknown type in component table entry

Symptom

An attempt to boot a Linux volume that is prepared for secure boot fails, and a message similar to the following is displayed on the console:

```
MLOEVL012I: Machine loader up and running (version v3.1.4).
MLOLOA027E: Unknown type in component table entry.
MLOEVL010E: IPL failed.
```

Problem

The hardware or hypervisor environment does not support secure boot. See “Supported environments” on page 1 for a list of hardware and software requirements for secure boot.

Solution

Reinstall the Linux boot volume without secure boot support. For details, see “Preparing a boot device” on page 5.

No signed components found

Symptom

An attempt to boot a Linux volume with secure boot enabled fails, and a message similar to the following is displayed on the console:

```
MLOLOA62693212 Audit: No signed components found for program 0
loaded from device
```

Problem

The Linux boot volume is not correctly prepared for secure boot - it is missing signature data. Signature data can be missing due to one of the following reasons:

- The Linux version does not support secure boot.
- The Linux boot-file signatures have been removed.
- No secure boot signatures were written during boot loader installation.

Solution

- Check the list in “Supported environments” on page 1 to ensure that the Linux version that is used supports secure boot.
- Reinstall Linux packages that contain boot files. For more information, see “Reinstalling Linux boot-file packages” on page 20.
- Reinstall the boot volume while forcing the installation of secure boot data. For more information, see “Preparing a boot device” on page 5.

Accessibility

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

Documentation accessibility

The Linux on IBM Z and IBM LinuxONE publications are in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when you use the PDF file and want to request a Web-based format for this publication send an email to eservdoc@de.ibm.com or write to:

IBM Deutschland Research & Development GmbH
Information Development
Department 3282
Schoenaicher Strasse 220
71032 Boeblingen
Germany

In the request, be sure to include the publication number and title.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility at

```
www.ibm.com/able
```

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.

SC34-7755-00