Need for Trustworthy Systems

In Cisco IOS XR Release 7.0.1, this section is applicable *only* to the following Cisco NCS 540 variants:

- N540-28Z4C-SYS-A/D
- N540X-16Z4G8Q2C-A/D
- N540-12Z20G-SYS-A/D
- N540X-12Z16G-SYS-A/D

Global service providers, enterprises, and government networks rely on the unimpeded operation of complex computing and communications networks. The integrity of the data and IT infrastructure is foundational to maintaining the security of these networks and user trust. With the evolution to anywhere, anytime access to personal data, users expect the same level of access and security on every network. The threat landscape is also changing, with adversaries becoming more aggressive. Protecting networks from attacks by malevolent actors and from counterfeit and tampered products becomes even more crucial.

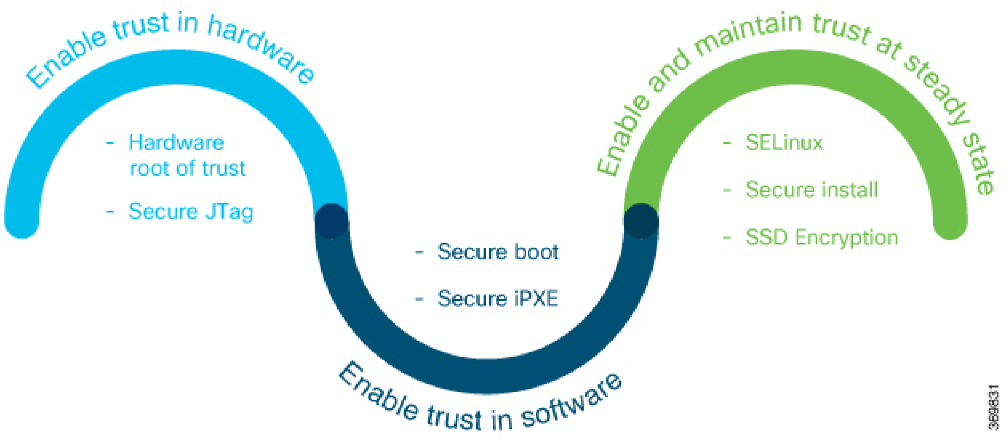
Routers are the critical components of the network infrastructure and must be able to protect the network and report on system integrity. A “trustworthy solution” is one that does what it is *expected* to do in a *verifiable* way. Building trustworthy solutions requires that security is a primary design consideration. Routers that constitute trustworthy systems are a function of security, and trust is about preventing as well as knowing whether systems have been tampered with.

In trustworthy systems, trust starts at the lowest levels of hardware and is carried through the boot process, into the operating system (OS) kernel, and finally into runtime in the OS.

The main components of implementing a trustworthy system are:

- Enabling trust in hardware with Hardware root-of-trust and secure JTag
- Enabling trust in software with secure boot and secure iPXE
- Enabling and maintaining trust at steady state with SELinux, Secure install, and SSD Encryption

Figure 1: Ecosystem of Trustworthy Systems



Trustworthy systems must have methods to securely measure hardware, firmware, and software components and to securely attest to these secure measurements.

For information on key concepts used in this chapter, see the Understanding Key Concepts in Security.

Enable Trust in Hardware

Because software alone can't prove a system's integrity, truly establishing trust must also be done in the hardware using a hardware-anchored root of trust. Without a hardware root of trust, no amount of software signatures or secure software development can protect the underlying system from becoming compromised. To be effective, this root of trust must be based on an immutable hardware component that establishes a chain of trust at boot-time. Each piece of code in the boot process measures and checks the signature of the next stage of the boot process before the software boots.

A hardware-anchored root of trust is achieved through:

- Anti-counterfeit chip: All modules that include a CPU, as well as the chassis, are fitted with an anti-counterfeit chip, which supports co-signed secure boot, secure storage, and boot-integrity-visibility. The chip ensures that the device's software and hardware are authentic and haven't been tampered with or modified in any way. It also helps to prevent unauthorized access to the device's sensitive data by enforcing strong authentication and access control policies.

- Secure Unique Device Identifier (SUDI): The X.509 SUDI certificate installed at manufacturing provides a unique device identifier. SUDI helps to enable anti-counterfeit checks along with authentication and remote provisioning. The SUDI is generated using a combination of the device's unique hardware identifier (such as its serial number or MAC address) and a private key that is securely stored within the device. This ensures that each SUDI is unique and cannot be easily duplicated or forged. When a device attempts to connect to a network, the network uses the SUDI to authenticate the device, and ensure that it's authorized to connect. This helps to prevent unauthorized access to the network and ensures that only trusted devices are allowed to connect.

- Secure JTag: The secure JTAG interface is used for debugging and downloading firmware. This interface with asymmetric-key based authentication and verification protocols prevents attackers from modifying firmware or stealing confidential information. Secure JTAG typically involves a combination of hardware and software-based security measures. For example, it may include the use of encryption and authentication protocols to secure communications between the JTAG interface and the debugging tool. It may also involve the use of access control policies and permissions to restrict access to the JTAG interface to authorized users only.

Note Hardware-anchored root of trust is enabled by default on Cisco NCS 540 Series routers.

Secure Hardware for Strong Cryptography

All Cisco IOS XR7 supported-platforms ships with a non-tamperable Trust Anchor module (TAm) in the hardware.

TAm houses known-good-values (KGVs) of the hardware components along with keys and certificates rooted to Cisco, which are used to verify components of the hardware during the BIOS boot.

Chip Guard and Attestation are security features implemented in TAm.

- Chip Guard detects tampering attempts and responds by initiating actions such as disabling access to the device, erasing sensitive information stored in the device, or triggering a security alarm.
- Attestation provides a mechanism for verifying the integrity and authenticity of the software and hardware components of a device.

A Cisco router with SUDI is authenticated and verified remotely for uniquely identifying that it’s an authentic Cisco device.

Where Cisco NCS 540 Series Routers have the older generation of chips with lesser capabilities compared to the latest TAm chips present on the newer generation of hardware.

Chip Guard

Attacks can come from various sources – starting from the CPUs or ASICs containing Trojan or malware. Sometimes, the chips may have a Trojan in form of an added die in the package assembly.

Cisco’s Chip Guard feature mitigates this threat with the use of unique identifiers buried inside the Trusted Anchor module (TAm) devices as a way to identify and track components throughout the lifecycle of products.

During the manufacturing phase, hashes of known good values (KGV) of the components are burnt into the TAm. At every boot of the system, the components are validated by matching their observed values with the KGV of that component present on the TAm.

If the values do not match, the system fails to boot up. In which case, the operator must perform a remote attestation to query the TAm and identify the cause of bootup failure.

Attestation

Proof of hardware integrity is recorded in the TAm as part of Chip Guard. This proof is made available through the following commands:

Note The same data is also available through NETCONF for a remote attestation server: Cisco-IOS-XR-remote-attestation-act.yang.

```
RP/0/RP0/CPU0:ios# show platform security ?
attest Attest the information
health Match and report any inconsistencies in secure variables across nodes
integrity System Integrity
tam Tam Device Details
variable Show secure variables from secure certificate storage

RP/0/RP0/CPU0:ios# show platform security integrity ?
dossier Ask me anything dossier
hardware Fetch System Hardware integrity
log Integrity Logs

RP/0/RP0/CPU0:ios# show platform security attest ?
PCR PCR quotes and value
certificate Fetch System Certificates

RP/0/RP0/CPU0:ios# show platform security attest PCR ?
WORD PCR register number. Specify multiple PCRs seperated by ','

RP/0/RP0/CPU0:ios# show platform security attest PCR 0 ?
location Certificates from which location
trustpoint CiscoSUDI/CiscoAIK to be used for PCR quote
| Output Modifiers

RP/0/RP0/CPU0:ios# show platform security attest PCR 0 location ?
0/RP0/CPU0 Fully qualified location specification
WORD Fully qualified location specification
all all locations

RP/0/RP0/CPU0:ios# show platform security attest PCR 0 location 0/RP0/CPU0 trustpoint locaion
 0/RP0/CPU0 tr
CiscoAIK Cisco AIK Certificate
CiscoSUDI Cisco SUDI Certificate

RP/0/RP0/CPU0:ios# show platform security attest certificate ?
CiscoAIK Cisco AIK Certificate
CiscoSUDI Cisco SUDI Certificate
```

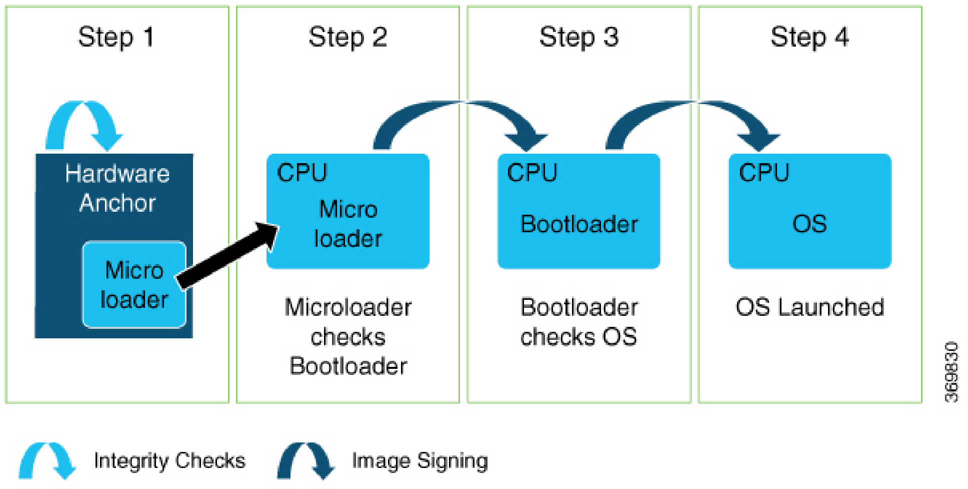
Enable Trust in Software

In Cisco IOS XR7, trust in the software is enabled through:

Secure Boot

Cisco Secure Boot helps to ensure that the code that executes as part of the software image boot up on Cisco routers is authentic and unmodified. Cisco IOS XR7 platforms support the hardware-anchored secure boot which is based on the standard Unified Extensible Firmware Interface (UEFI). This UEFI-based secure boot protects the microloader (the first piece of code that boots) in tamper-resistant hardware, establishing a root of trust that helps prevent Cisco network devices from executing tainted network software.

Figure 2: Secure Boot



The intent of Secure Boot is to have a trust anchor module (TAm) in hardware that verifies the bootloader code. A fundamental feature of secure boot is the barrier it provides that makes it that it is extremely difficult or nearly impossible to bypass these hardware protections.

Secure boot ensures that the bootloader code is a genuine, unmodified Cisco piece of code and that code is capable of verifying the next piece of code that is loaded onto the system. It is enabled by default.

When secure boot authenticates the software as genuine Cisco in a Cisco device with the TAm, the operating system then queries the TAm to verify whether the hardware is authentic. It verifies by cryptographically checking the TAm for a secure unique device identifier (SUDI) that comes only from Cisco.

The SUDI is permanently programmed into the TAm and logged by Cisco during Cisco's closed, secured, and audited manufacturing processes.

Booting the System with Trusted Software

In Cisco IOS XR7, the router supports the UEFI-based secure boot with Cisco-signed boot artifact verification. The following takes place:

Step 1: At bootup, the system verifies every artifact using the keys in the TAm.

Step 2: The following packages are verified and executed:

- Bootloader (Grand Unified Bootloader (GRUB), GRUB configuration, Preboot eXecution Environment (PXE), netboot)
- Initial RAM disk (Initrd)
- Kernel (operating system)

Step 3: Kernel is launched.

Step 4: Init process is launched.

Step 5: All Cisco IOS XR RPMs are installed with signature verification.

Step 6: All required services are launched.

Secure iPXE – Secure Boot Over the Network

The iPXE server is an HTTP server discovered using DHCP that acts as an image repository server. Before downloading the image from the server, the Cisco router must authenticate the iPXE server.

Note A secure iPXE server must support HTTPS with self-signed certificates.

The Cisco router uses certificate-based authentication to authenticate the iPXE server. The router:

- Downloads the iPXE self-signed certificates
- Uses the Simple Certificate Enrollment Protocol (SCEP)
- Acquires the root certificate chain and checks if it’s self-signed

The root certificate chain is used to authenticate the iPXE server. After successful authentication, a secure HTTPS channel is established between the Cisco router and the iPXE server. Bootstrapper protocol (Bootp), ISO, binaries, and scripts can now be downloaded on this secure channel.

Establish and Maintain Trust at Steady State

Attackers are seeking long-term compromise of systems and using effective techniques to compromise and persist within critical infrastructure devices. Hence, it is critical to establish and maintain trust within the network infrastructure devices at all points during the system runtime.

In Cisco IOS XR7, trust is established and maintained in a steady state through:

SELinux

Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides a mechanism for supporting access control security policies, including mandatory access controls (MAC).

A kernel integrating SELinux enforces MAC policies that confine user programs and system servers to the minimum amount of privileges they require to do their jobs. This reduces or eliminates the ability of these programs and daemons to cause harm when compromised (for example, through buffer overflows or misconfigurations). This confinement mechanism operates independently of the traditional Linux access control mechanisms. SELinux has no concept of a "root" super-user and does not share the well-known shortcomings of the traditional Linux security mechanisms (such as a dependence on setuid/setgid binaries).

On Cisco IOS XR7 software, only Targeted SELinux policies are used, so that only third-party applications are affected by the policies; all Cisco IOS XR programs can run with full root permission.

With Targeted SELinux, using targeted policies, processes that are targeted run in a confined domain. For example, the httpd process runs in the httpd_t domain. If a confined process is compromised by an attacker, depending on the SELinux policy configuration, the attacker's access to resources and the possible damage that can result is limited.

Note Processes running in unconfined domains fall back to using discretionary access control (DAC) rules.

DAC is a type of access control defined as a means of restricting access to objects based on the identity of the subjects or the groups (or both) to which they belong.

Confined and Unconfined Users

Each Linux user is mapped to an SELinux user through an SELinux policy. This allows Linux users to inherit the restrictions placed on SELinux users.

If an unconfined Linux user executes an application, which an SELinux policy defines as an application that can transition from the unconfined_t domain to its own confined domain, the unconfined Linux user is subject to the restrictions of that confined domain. The security benefit is that, even though a Linux user is running in unconfined mode, the application remains confined. Therefore, the exploitation of a flaw in the application is limited by the policy.

A confined Linux user is restricted by a confined user domain against the unconfined_t domain. The SELinux policy can also define a transition from a confined user domain to its own target confined domain. In such a case, confined Linux users are subject to the restrictions of that target confined domain.

SELinux Mode

There are three SELinux modes:

- Enforcing: When SELinux is running in enforcing mode, it enforces the SELinux policy and denies access based on SELinux policy rules.

  SELinux security policy is configured as *enforcing* by default.

- Permissive: In permissive mode, the SELinux does not enforce policy, but logs any denials. Permissive mode is used for debugging and policy development.

- Disabled: In disabled mode, no SELinux policy is loaded. The mode may be changed in the boot loader, SELinux config, or at runtime with **setenforce**.

To view security policy mode:

```
[node0_RP0_CPU0:~]$getenforce
Enforcing
```

Role of the SELinux Policy in Boot Process

SELinux plays an important role during system startup. Because all processes must be labeled with their proper domain, the init process performs essential actions early in the boot process that synchronize labeling and policy enforcement.

After the kernel is loaded during boot, the initial process is assigned the predefined initial SID kernel_t. Initial SIDs are used for bootstrapping before the policy is loaded. The init process scans the /etc/selinux/config directory for the active policies, such as the targeted policy, and loads the associated file.

After the policy is loaded, the initial SIDs are mapped to security contexts in the policy. In the case of the targeted policy, the new domain is "user_u:system_r:unconfined_t". The kernel begins to get security contexts dynamically from the in-kernel security server.

The init process then re-executes itself so that it can transition to a different domain, if the policy defines it. For the targeted policy, there is no transition defined and the init process remains in the unconfined_t domain. At this point, the init process continues with its normal boot process.

Secure Install

The Cisco IOS XR software is shipped as RPMs. Each RPM consists of one or more processes, libraries, and other files. An RPM represents a collection of software that performs a similar functionality; for example, packages of BGP, OSPF, as well as the Cisco IOS XR Infra libraries and processes.

RPMs can also be installed into the base Linux system outside the Cisco IOS XR domain; however, those RPMs must also be appropriately signed.

All RPMs shipped from Cisco are secured using digitally signed Cisco private keys.

There are three types of packages that can be installed:

- Packages shipped by Cisco (open source or proprietary)
- Customer packages that replace Cisco provided packages
- Customer packages that do not replace Cisco provided packages

RPM Signing and Validation

RPMs are signed during the build process, when the different RPMs are "constructed" using the packaging instructions of the build process. Any package - process, library, or file - can exist in only one RPM. For example, if BGP is packaged as a separate RPM, then any artifacts related to BGP are present only in the BGP RPM and not, for example, in the Routing RPM.

The install component of the Cisco IOS XR performs various actions on the RPMs, such as verification, activation, deactivation, and removal. Many of these actions invoke the underlying DNF installer. During each of these actions, the DNF verifies the signature of the RPM to ensure that it operates on a legitimate package.

X.509 Certificates for RPM Signing

- X.509 certificates provide a single way to manage the system's certificates for verification, delegation, rollover, revocation, policy control, and so on.
- X.509 offer higher flexibility than other certificate formats.

Note The X.509 certificate used to sign the RPM must be pulled in from the TAm into the kernel key ring, along with the rest of the keys.

Modifying the RPM Header

The RPM certificate keys are taken out during the boot process and added into the kernel keyring by kernel patches from the UEFI. During the run time of Cisco IOS XR7 software, these keys are always present in the kernel keyring. The RPM metadata signature header can be modified to specify that the key type is a kernel keyring-based key. When the RPM needs to be validated, RPM executable picks the key from the kernel keyring to validate it.

Note The signature type in the RPM and during the build continue to be GPG based.

Third-Party RPMs

The XR Install enforces signature validation using the 'gpgcheck' option of DNF. Thus, any Third-Party RPM packages installation fails if done through the XR Install (which uses the DNF). However, Third-Party RPMs can still be installed using the **rpm** command.

Boot Integrity and Trust Visibility

The secure boot first stage is rooted in the chip and all subsequent boot stages are anchored to the first successful boot. The system is, therefore, capable of measuring the integrity of the boot chain. The hash of each software boot image is recorded before it is launched. These integrity records are protected by the TAm. The boot chain integrity measurements are logged and these measurements are extended into the TAm.

Use the **show platform boot-integrity [sign [nonce <nonce>] [trustpoint <AIK trustpoint name>]]** command to view the boot integrity and boot-chain measurements.

You can also use Cisco-IOS-XR-remote-attestation-act.yang to fetch the boot integrity over the NETCONF protocol.

The command displays both, the integrity log values and the assurance that these values have not been tampered. These measurements include the following parameters:

- Micro loader hash
- Boot loader hash
- Image signing and management key hashes
- Operating system image hash

```
platform-pid string Platform ID
Event log [key: event_number]: Ordered list of TCG described event log
                               that extended the PCRs in the order they
                               were logged
    +-- event_number   uint32 Unique event number of this even
    +-- event_type     uint32 log event type
    +-- PCR_index      uint16 PCR index that this event extended
    +-- digest         hex-string The hash of the event data
    +-- event_size     uint32 Size of the event data
    +-- event_data     uint8[] event data, size determined by event_size
PCR [index] - List of relevant PCR contents
    +-- index    uint16 PCR register number
    +-- value   uint8[] 32 bytes - PCR register content
PCR Quote binary TPM 2.0 PCR Quote
PCR Quote Signature binary Signature of the PCR quote using TAM-held ECC  or RSA restricted
 key with the optional nonce if supplied
```

Note

- PCR 0-9 are used for secure boot.
- Signature version designates the format of the signed data.
- The signature digest is SHA256.
- The signing key is in a TCG compliant format.

```
RP/0/RP0/CPU0:ios# show platform security attest PCR 15 trustpoint CiscoAIK nonce 4567
location 0/RP0/CPU0

Sun Jun 21 03:07:18.394 UTC
Nonce: 4567

+---------------------------------------+
   Node location: node0_RP0_CPU0
+---------------------------------------+
Uptime: 495270
pcr-quote:
/1RDR4AYACCBG/wltf4IEwfdUjtjun7S3rXC90eAb0G0ytrYRv3ExwACRWcAAAAAAD8hUwAAAEf/////AQAAACQAAAAIAAAAAQAIAwCAAAAgae1J8QIYe06nS2RUx0JYeoG8tM3oqeVdpW7COowBt+g=
pcr-quote-signature:
[illegible]
pcr-index       pcr-value
  15    oYk8yqrzudIpGB4H++SaV0wMv6ugDSUIuUfeSqbJvbY=

RP/0/RP0/CPU0:ios# show platform security integrity hardware digest-algorithm SHA1 trustpoint
 CiscoAIK nonce 4567 location 0/RP0/CPU0

Sun Jun 21 03:09:14.594 UTC
Nonce: 4567

+---------------------------------------+
   Node location: node0_RP0_CPU0
+---------------------------------------+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495385
Known-good-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
observed-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
PCRs:
Index   value
  15    1Y3uKqNv1UJQUNZQxmZkiuG4blk=

RP/0/RP0/CPU0:ios# show platform security integrity hardware digest-algorithm SHA256
trustpoint CiscoAIK nonce 4567 location 0/RP0/CPU0

Sun Jun 21 03:09:31.110 UTC
Nonce: 4567

+---------------------------------------+
   Node location: node0_RP0_CPU0
+---------------------------------------+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495401
Known-good-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
observed-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
PCRs:
Index   value
  15    1Y3uKqNv1UJQUNZQxmZkiuG4blk=

RP/0/RP0/CPU0:ios# show platform security integrity hardware digest-algorithm SHA256
trustpoint CiscoAIK nonce 4567 location 0/RP0/CPU0

Sun Jun 21 03:09:43.782 UTC
Nonce: 4567
```

```
+--------------------------------------+
   Node location: node0_RP0_CPU0
+--------------------------------------+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495414
Known-good-digests:
Index   value
  0     y3n/SsvyNb8g3o7FFRGCZwfbs8EGxvMZg/PeN0NA71k=
observed-digests:
Index   value
  0     y3n/SsvyNb8g3o7FFRGCZwfbs8EGxvMZg/PeN0NA71k=
PCRs:
Index   value
  15    oYk8yqrzudIpGB4H++SaV0wMv6ugDSUIuUfeSqbJvbY=
Cisco AIK Certificate used for signing PCR
pcr-quote:
/1RDR4AYACBG/wltf4TEwfdUjtjun7S3rXC90eAb0G0ytrYRv3ExwACRWcAAAAAAD8hywAAAEf/////AQAAACQAAAALAAAAAQALAwCAAAAgae1J8QIYe06nS2RUx0JYeoG8tM3oqeVdpW7CObwBt+g=
pcr-quote-signature:
[illegible]
RP/0/RP0/CPU0:ios#show platform security integrity hardware digest-algorith$
Sun Jun 21 03:09:56.794 UTC
Nonce: 4567

+--------------------------------------+
   Node location: node0_RP0_CPU0
+--------------------------------------+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495427
Known-good-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
observed-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
PCRs:
Index   value
  15    1Y3uKqNv1UJQUNZQxmZkiuG4blk=

RP/0/RP0/CPU0:ios#
```

Note Use the **show platform security tam** command to view the TAm device details.

Boot integrity verification consists of the following steps:

1. Report Boot 0 version and look up the expected integrity value for this platform and version.
2. Report bootloader version and look up the expected integrity value for this platform and version.
3. Report OS version and look up the expected integrity value for this platform and version.
4. Using the integrity values obtained from steps 1-3, compute the expected PCR 0 and PCR 8 values
5. Compare the expected PCR values against the actual PCR values.
6. Verify the nonced signature to ensure the liveliness of the response data.
7. (Optional) Verify the software image (IOS XR) version is with what is expected to be installed on this platform.

A failure of any of the above steps indicates either a compromised system or an incomplete integrity value database.

Secure gRPC

gRPC (gRPC Remote Procedure Calls) is an open source remote procedure call (RPC) system that provides features such as, authentication, bidirectional streaming and flow control, blocking or nonblocking bindings, and cancellation and timeouts. For more information, see https://opensource.google.com/projects/grpc.

TLS (Transport Layer Security) is a cryptographic protocol that provides end-to-end communications security over networks. It prevents eavesdropping, tampering, and message forgery.

In Cisco IOS XR7, by default, TLS is enabled in gRPC to provide a secure connection between the client and server.

Integrity Measurement Architecture (IMA)

The goals of the Linux kernel integrity subsystem are to:

- detect whether files are accidentally or maliciously altered, both remotely and locally
- measure the file by calculating the hash of the file content
- appraise a file's measurement against a known good value stored as an extended attribute
- enforce local file integrity

Note These goals are complementary to the Mandatory Access Control (MAC) protections provided by SElinux.

IMA maintains a runtime measurement list and—because it is also anchored in the hardware Trusted Anchor module (TAm)—an aggregate integrity value over this list. The benefit of anchoring the aggregate integrity value in the TAm is that the measurement list cannot be compromised by any software attack without being detectable. As a result, on a trusted boot system, IMA-measurement can be used to attest to the system's runtime integrity.

For more information about IMA, download the IMA whitepaper, An Overview of The Linux Integrity Subsystem.

IMA Signatures

The IMA appraisal provides local integrity, validation, and enforcement of the measurement against a known good value stored as an extended attribute—security.ima. The method for validating file data integrity is based on a digital signature, which in addition to providing file data integrity also provides authenticity. Each file (RPM) shipped in the image is signed by Cisco during the build and packaging process and validated at runtime using the IMA public certificate stored in the TAm.

All RPMs contain Cisco IMA signatures of the files packaged in the RPM, which are embedded in the RPM header. The IMA signature of the individual file is stored in its extended attribute during RPM installation. This protects against modification of the Cisco RPMs.

The IMA signature format used for IMA can have multiple lines and every line has comma-separated fields. Each line entry will have the filename, hash, and signature as illustrated below.

- File – Filename with the full path of the file hashed and signed
- Hash – SHA256 hash of the file
- Signature – RSA2048 key-based signature

How Trustworthiness is Implemented

The following sequence of events takes place when the Cisco routers that support IOS XR7 operating system are powered up:

1. At power UP, the micro-loader in the chip verifies the digital signature of BIOS using the keys stored in the TAm. The BIOS signature verification is logged and the measurement is extended into a PCR.
2. The BIOS then verifies the signature of the boot-loader using keys stored in TAm. The boot-loader signature verification is logged and the measurement is extended into the PCR.
3. If the validation is successful, the BIOS launches the bootloader. The bootloader uses the keys loaded by the BIOS to verify the sanctity of the kernel, initrd file system, and grub-config file. Each verification operation is logged, and the PCR in TAm is extended.
4. The initrd is exploded to create the initial file system.
5. The kernel is launched and the kernel keyrings are populated with the appropriate keys from the TAm.
6. Kernel modules are verified. Module verification results are logged and TAm PCR is extended.
7. The init process is launched. Whenever an executable or a shared library is invoked, the IMA kernel hook validates the signature using the certificates in IMA keyring, which is then used to validate the signature attached to the file.
8. The Cisco IOS XR7 RPM is installed with the signed verification. The results of RPM verification are logged.
9. Cisco IOS XR7 processes are launched with IMA measurement.
10. TAm services are launched.
11. Cisco IOS XR7 application runs the initial admin user configuration and stores the credentials into TAm secure storage.

    Manual provisioning of user credentials is now complete.

After the sequence is successfully completed, the router is considered trustworthy.

Understanding Key Concepts in Security

Attestation

Attestation is a mechanism used to attest the software's integrity. The verifier trusts that the attested data is accurate because it is signed by a TPM whose key is certified by the CA.

Attestation Identity Key

An Attestation Identity Key (AIK) is a restricted key that is used for signing attestation requests.

Bootloader

The bootloader is a piece of code that runs before any operating system begins to run. Bootloaders contain several ways to boot the OS kernel and also contain commands for debugging and modifying the kernel environment.

Certificates and Keys in TAm

All database keys are signed by the KEK. Any update to the keys requires the KEK or PK to sign in, using time-based authentic variables. Some of the keys on the database are:

- Image signing certificate: This is the X.509 certificate corresponding to the public key and is used for validating the signature of grub, initrd, kernel, and kernel modules.
- IOS-XR Key: A public key certificate signed by the KEK. This key is common to all Cisco NCS 540 Series routers and is used to sign GRUB, initrd, kernel and kernel modules.
- RPM key: Used for signing RPMs.
- IMA public key certificate: Used for Integrity Measurement Architecture (IMA), and used to validate the IMA signature of the files.
- BIOS or Firmware Capsule Update key: Used to sign the outer capsule for BIOS or firmware updates. It is the same as the secure boot key.
- Platform key (PK) and Key Enrollment Key (KEK): These are public keys and certificates used to manage other keys in the TAM.
- LDWM Key: In the Cisco IOS XR7, the LDWM key is stored in the hardware trust anchor module and is used for validating the BIOS.

Golden ISO (GISO)

A GISO image includes a base binary artifact (an ISO) for the Linux distribution that is used on the server fleet, packages, and configuration files that can be used as a base across all servers.

The GISO image for Cisco IOS XR7 software contains the IOS XR RPMs and third-party RPMs.

GRand Unified Bootloader (GRUB)

GNU GRUB (or just GRUB) is a boot loader package that loads the kernel and supports multiple operating systems on a device. It is the first software that starts at a system boot.

Hash Function

A hash function is any function that is used to map data of arbitrary size onto data of a fixed size.

Initramfs

Initramfs, a complete set of directories on a normal root filesystem, is bundled into a single cpio archive and compressed with one of the several compression algorithms. At boot time, the boot loader loads the kernel and the initramfs image into memory and starts the kernel.

initrd

initial RAM disk is an initial root file system that is mounted before the real root file system is made available. The initrd is bound to the kernel and loaded as part of the kernel boot procedure.

JTAG

JTAG is a common hardware interface that provides a system with a way to communicate directly with the chips on a board. JTAG is used for debugging, programming, and testing on embedded devices.

Nonce Value

A nonce value is an arbitrary number that can be used only once in a cryptographic communication. It is a random or pseudo-random number that is issued in an authentication protocol to ensure that the old communications are not reused in replay attacks.

Platform Configuration Register (PCR)

PCR is a 256-bit storage location for discrete integrity measurements. It is designed to hold an unlimited number of measurements in the register. It does this by using a cryptographic hash and hashing all updates to a PCR.

Trust Anchor module (TAm)

The Cisco Trust Anchor module (TAm) helps verify that Cisco hardware is authentic and provides additional security services.