Protecting Secrets on Cisco Network Devices

Last updated March 12, 2026 – Initial Release for IOS XE

Cisco network devices rely on a variety of secrets (passwords, pre-shared keys, etc.) to secure communications with other devices or users. For example, an administrator may configure a username and password that can be used to log in to the device or a pre-shared key to authenticate a connection to a TACACS+ or RADIUS server.

Secrets generally fall into two categories: reversible and non-reversible credentials.

A non-reversible credential is one where the device does not need to know the secret but needs to be able to verify that a far-end user/device does know the secret. For example, when a user authenticates a management connection via SSH or HTTPS, the network device must be able to determine whether the remote user knows the password associated with the account when the remote user provides the password. The hash cannot be reversed to the original password but can be used to determine whether someone in possession of the secret knows what it is.

A reversible credential is one where the device needs to be able to derive the original secret because it needs to use the original plain text version of the secret for some purpose like routing protocol authentication or TACACS+ / RADIUS pre-shared keys.

Cisco network devices have a variety of mechanisms available for storing both reversible and non-reversible credentials. These mechanisms have evolved over time and have varying levels of security (or lack thereof) associated with them. This document describes the various mechanisms that exist and forthcoming changes aimed to increase the security posture of secrets stored on Cisco network devices.

Cisco network devices (IOS XE, IOS XR, NX-OS) have numbered “types” associated with each kind of credential. The following table lists the various credential types:

Type	Reversibility	Description	Secure
0	n/a	Plain Text – Unencrypted	✗
4	Non-Reversible	Weak implementation of SHA256	✗
5	Non-Reversible	MD5	✗
6	Reversible	AES128 Encryption using master key	✓
7	Reversible	Vigenère Cipher	✗
8	Non-Reversible	PBKDF2, SHA-256, 80-bit salt, 20,000 iterations	✓
9	Non-Reversible	SCRYPT (RFC 7914), 80-bit salt, 16,384 iterations	✓
10	Non-Reversible	PBKDF2-HMAC-SHA512 (IOS XR Only)	✓

Unencrypted (Type 0)

Type 0 credentials are not encrypted or hashed in any way. The credential is shown in plain text in the configuration file and therefore exposes the credential to anyone with access to the configuration file. **Type 0** credentials may appear in configuration for features that require either reversible or non-reversible credentials. When configuring a device, a credential is typically provided in plain text format. Depending on the configuration of the device, this plain text credential is either stored as-is or converted to one of the other types before being stored in the configuration file, depending on the feature being configured. **Type 0** *credentials should never be used in running configuration file on a device, as it will expose credentials to anyone who can gain access to the configuration file*.

Non-Reversible Credentials

Non-reversible credentials using cryptographic one-way functions (hashes) where the network device only stores the result of the one-way function, not the secret itself. This means the device cannot derive the original plain text. This is typically used for user credentials where the device only needs to verify that the far-end (e.g. the user logging in) knows the secret.

Type 4 credentials are mentioned here for the sake of completeness, but they were only available for a limited number of releases and deprecated immediately (for more information, see this security advisory.) They should never be used and are no longer available in any Cisco network operating system.

Type 5 credentials use MD5, an algorithm that has many weaknesses that have been well known for over a decade, mainly collision attacks and the low compute requirements make MD5 more susceptible to brute force and dictionary attacks. MD5 is considered weak and should not be used to store sensitive data.

Types 8 (sha256), **9** (scrypt), and **10** (sha512) are all considered secure credential types that use strong cryptographic hashing functions and many iterations to make cracking very difficult. **Type 9** is generally considered stronger than **type 8**, but some customers choose **type 8** for regulatory reasons.

Once a non-reversible credential is configured on a device, this configuration is portable from one device to another. For example, the following line configures the username test on an IOS XE device:

```
username test secret test
```

This results in the following configuration displayed in the running configuration file:

```
username test secret 9
$9$VwY2Qzmm29EQ3U$E0hNsbytLx6lLr/A.e0tUs4ofSPb47YN0odBf6kv4Zs
```

If the above configuration with the **type 9** credential is configured on another IOS XE device, it will create the same user with a username of 'test' and a password of 'test'. There are no external dependencies because these are one-way hashes and there is no way for someone who obtains the **type 9** credential to be able to reverse it back to the password of 'test'. The configuration line includes both the salt as well as the hash, so every time the password "test" is configured, the output will look completely different because the salt will always be different. This ensures that traditional rainbow table attacks are not possible.

Beginning with release 16.12.x of IOS XE, **type 5** credentials are automatically converted to **type 9** to enhance the security of configuration files. You may be wondering how this is accomplished if the original credential is a one-way hash. The Cisco credential stored contains information indicating that it is a **type 5** credential that was converted to **type 9**. The **type 9** algorithm is used to hash the **type 5** hash that was previously stored. When attempting to validate a credential, the device knows it needs to first perform an MD5 hash followed by a PBKDF2 hash to validate the credential. This is all handled automatically and is transparent to the end user.

For example, if a user has configured a password of "test" as a **type 5** credential, it gets stored as an MD5 hash of the word "test" – let's say <md5_hash_of_test>. Older versions of IOS XE would display < md5_hash_of_test > in the configuration file. When upgrading to a version that supports automatic conversion, the operating system will apply the **type 9** algorithm (scrypt w/ 80-bit salt over 16k iterations) and come up with an scrypt hash of the MD5 hash. Let's call this <scrypt_hash_of_md5_hash_of_test>. This new value will be stored in the configuration file as a **type 9** credential with an indication that this is a hash of something that was previously an MD5 hash. When a user logs into the device and presents the password of "test", the operating system recognizes that it needs to apply this special procedure, so instead of immediately performing a hash of "test" (using the same 80-bit salt that is in the configuration file), it first performs an MD5 hash of "test" to arrive at <md5_hash_of_test>, then applies the scrypt algorithm (again using the 80-bit salt in the configuration) to the result of the MD5 operation to arrive at <scrypt_hash_of_md5_hash_of_test>. It then compares this value with what is in the configuration and if they match, the password is validated. At no point can the device derive the word "test" from the configuration, but if someone presents the word "test" as a credential, the device can validate that it is indeed the expected value.

Reversible Credentials

In cases where a feature or protocol requires that the network device be able to obtain the unencrypted version of a secret, the device must use a reversible credential. The most basic form of a reversible credential is plain text. When a secret is configured in plain text, the device has immediate access to the secret, however this also means that an attacker who obtains access to the configuration file will also be able to see all the secrets in plain text.

For decades, Cisco network devices have supported 'password-encryption,' but this configuration generates weak **type 7** credentials using the Vigenère cipher. The cipher itself is centuries old and very weak. Additionally, **type 7** credentials use a static key that is embedded in the Cisco operating system. This key was publicly discovered decades ago and is widely published in security research. This means that **type 7** credentials should, for all intents and purposes, be treated like plain text. They offer no security beyond perhaps protecting against a casual shoulder surfing attack where the secrets are not immediately obvious upon a quick glance at a configuration file. Because **type 7** credentials use the same encryption key and algorithm on all devices, a configuration with either a plain text (**type 0**) or **type 7** credential is portable to another device because any device (or attacker for that matter) can easily decode the credential. A configuration with a **type 7** credential can be copied to another device and the new device will have access to all the secrets.

In 2006, Cisco introduced the **type 6** credential to secure credentials stored in device configuration files. **Type 6** uses AES 128 encryption to encrypt nonreversible credentials like pre-shared keys and some passwords. On IOS XE, the use of **type 6** credentials is enabled with the global configuration 'password-encryption aes'; however, **type 6** credentials introduce an additional dependency. To perform **type 6** encryption, the device must first have an encryption key. Unlike **type 7** credentials that use a static key defined by Cisco, **type 6** uses keys that are unique to the device. The encryption key is configured using the 'key config-key' command. The value entered using this command is used as a seed to derive a 128-bit AES encryption key. This key can never be retrieved and is never shown in a configuration file. It must be kept secret by the administrator and stored independent of the configuration file to enable configuration portability as discussed below.

Because each device could (and should) have a unique encryption key, enabling **type 6** credentials introduces some operational overhead which is necessary to keep the secrets secure. Unlike the non-reversible credential types, configurations with **type 6** credentials have some restrictions on their portability. To copy a configuration with **type 6** credentials from one device to another, you must first configure the same encryption key on the new

device before inserting the configuration from the other device. If you know the encryption key and configure it on the device prior to inserting the configuration, the configuration is completely portable and will function as expected on the new device.

If you do not know the encryption key that was used to encrypt the configuration on the source device, you can still copy the configuration over to the new device, but you will have to re-provision any **type 6** credentials with their plain text versions as you did when you initially configured the device because the type 6 configuration will fail on the new device if the new device does not have the same key that was used to encrypt the original configuration.

Threat actors are increasingly exploiting the weakness of **type 0** and **type 7** credentials to gain access to sensitive credential data by obtaining device configuration files through a variety attacks. These attacks can range from simple use of known credentials to access the running configuration of a device, stealing flash cards from devices with the configuration stored on them, finding unauthenticated network servers (e.g. TFTP, HTTP) with configuration files, or other methods where configuration files might be harvested from an administrator's local personal computer, for example.

To reduce this risk for all customers, Cisco network devices will fully deprecate and remove the ability to store **type 0** or **type 7** credentials where reversible credentials are required. Cisco has been warning customers for years that type 7 credentials are deprecated and is now acting to fully remove support. Several significant changes will be made to achieve the final goal of removing support for type 7 credentials as follows. This document describes the process for IOS XE, but IOS XE, IOS XR, and NX-OS will follow similar plans, the specifics of which will be documented in the future.

1. Some features have not implemented support for **type 6** credentials and only accept **type 0** or **type 7** credentials in certain releases. Some of these features have alternative configuration options that do support **type 6** – for example some routing protocols support both native interface configuration and keychain configuration for routing protocol authentication, but only support **type 6** when using keychain. Features that do not support **type 6** will add support for **type 6** or will be deprecated in favor of the alternative configuration options that have support for **type 6**. For those features adding native support for **type 6** credentials, the support will be added in two phases to ensure downgrade scenarios are supported. In the first phase, features will add support for **type 6** credentials but will not automatically convert **type 0** or **type 7** credentials to **type 6**, even if an encryption key is configured. For IOS XE, this is planned for the IOS XE: 26.1.1release.

In a subsequent feature release (26.2.1 for IOS XE), all credentials will be converted to **type 6** automatically if an encryption key is present. Upgrade and downgrade scenarios are discussed in a separate section below.

2. As mentioned previously, an encryption key is required for **type 6** encryption to work. After upgrading to IOS XE 26.1.1, administrators will be strongly encouraged to configure a master key after upgrading if one has not been previously configured on the device.
   In this release, a customer will receive daily periodic syslog messages indicating that an encryption key is missing and must be configured. The device will continue to operate normally with unencrypted (or weakly obfuscated – type 7) credentials. The device will also display a warning after successful authentication to the device, and any time configuration mode is entered (i.e. 'config t'). The warning will also indicate that future upgrades will be blocked until an encryption key is configured. (with some exceptions as mentioned below in the downgrade section). Once a customer configures an encryption key, features supporting **type 6** credentials will be automatically encrypted (NOTE that IOS XE 26.1.1 also requires the command 'password-encryption aes' be configured, but releases 26.2.1 and later will encrypt automatically if an encryption key is present).
3. If a customer does not configure an encryption key prior to upgrading to IOS XE 27.1.1, the upgrade will fail until the customer configures an encryption key. Customers are strongly encouraged to configure an encryption key before attempting to upgrade to this releases to ensure a successful upgrade. Customers should securely store the encryption key in case they ever need to migrate the configuration to another device (for example in the case of an RMA).
4. Starting with the IOS XE 26.2.1 release, when a customer performs a new installation or erases the configuration on the device, an encryption key will be generated automatically upon boot of the device if one does not already exist. Upon boot, the customer *will be notified that a key has been automatically generated* and will be used to automatically encrypt any credentials that are configured on the device. There is no way for a customer to retrieve this encryption key and if configuration portability is not a requirement in the customer's environment, they can continue to use the auto-generated key without ever having to know what it is. If a customer would like to use their own key rather than using an auto-generated key, they must configure the encryption key on the device before applying any configuration. Changing the encryption key (without knowing the previous key) requires that all **type 6** credentials be removed from the configuration prior allowing for the encryption key to be changed, but in a new install scenario,

there are no credentials to invalidate, therefore this in effect allows the customer to change they key to one of their own choosing without any negative ramifications.

Customers who would like configuration portability between devices without having to re-enter the plain-text credentials in their configuration file must configure and manage their own encryption keys on their devices. This means ensuring that an encryption key is configured on the device before entering any credentials in the configuration. This also means ensuring that an encryption key is configured on the device prior to upgrading to a version will automatically convert configurations to **type 6**.

If a customer loses/forgets their encryption key or decides to proceed with the auto-generated key on a new installation, the device will function properly but will be limited in configuration portability options in the event they want to copy the configuration from one device to another device. To perform such a migration where the encryption key is not known (for example in the case of a device RMA or using a configuration as a template for another device) the customer must replace any **type 6** credentials in the configuration file with the plain-text versions of those credentials so that they may be re-encrypted on the new device. Backing up the configuration and restoring it on the same device will work even if the encryption key is unknown as long as a 'write erase' is not performed. Issuing a 'wr erase' command will erase the stored encryption key; however, the 'config replace' command can be used to safely replace the configuration with a backup copy without affecting the configured encryption key. Before performing a config replace, make sure that the correct master key is configured. The master key must match the one used to encrypt any passwords in the configuration to ensure seamless operation. If there is a mismatch between the master key and the encrypted configuration, the config replace operation will fail.

Given these restrictions, a customer has two options when attempting to migrate a configuration from one device to another:

- Configure the encryption key on devices and maintain a copy of it to make configurations portable between devices while maintaining the encrypted **type 6** credentials in the configuration file.
- Before moving the configuration to the new device, re-enter the plain-text credentials in the place of any **type 6** credentials in the configuration file (the customer must store the plain text versions of the credentials securely in an external credential store). The device will automatically convert these to new **type 6** credentials using the configured encryption key on the new device (whether manually set or automatically generated).

If a customer wants to know whether a device is using a user-supplied encryption key or an autogenerated key, they can issue the command **show crypto master-key-status**.

Reversible Credential Downgrade Scenarios

As mentioned previously, not all features have always supported **type 6** credentials. This means that once a device has been upgraded to a release that does support **type 6** credentials for a feature, it can only be safely downgraded to a version that also supports **type 6** credentials for that feature without losing that configuration upon downgrade. Note that previous migrations like this have occurred in the past such as with TACACS server keys.

To ensure customers have a viable downgrade path in the event of wanting to go back to a previous version, **type 6** credential support has been phased in as mentioned earlier. This strategy provides for one release train that supports **type 6** credentials but does not automatically convert **type 0** or **type 7** credentials to **type 6**. This release train can be used for direct downgrades as well as an interim step in the case the customer wants to downgrade to a release that does not have **type 6** support at all for that feature.

For example, if feature X on IOS XE does not support **type 6** credentials in release 17.18.x and support for **type 6** is introduced in the 26.1.x releases, upgrading from 17.18.x to 26.1.1 will not automatically convert the credentials for feature X to **type 6**, but a customer can manually convert if they so choose. Upon upgrading to 26.2.x, the credentials for feature X will be automatically converted to **type 6**. If a customer wants to downgrade to 26.1.x, they can do so safely because 26.1.x already has support for **type 6** credentials for feature X. If the customer wants to downgrade from 26.2.x to 17.18.x (or earlier), they must first downgrade to 26.1.x, change the **type 6** credential for feature X back to **type 0** or **type 7**, and then downgrade to 17.18.x.