ICSF Security Policies

ICSF provides a number of security controls that can provide additional security and protection related to cryptography. The traditional security around cryptographic resources relies on the System Authorization Facility and protects two types of resources: keys and APIs. That is, access to specific keys and specific APIs is checked as each resource is used and RACF is queried for whether the user is authorized to the resource.

The key security controls provide additional protections, some not necessarily specific to access lists. A good example is the Duplicate Key Token Check which simply prevents the insertion of records into the keystore if the key material matches a key that is already defined in the keystore. This control has nothing to do with access lists, however enabling it provides better security for key material, because there is only one set of access rules for a specific key. If two key records contain the same key value, one key record might be properly secured, while the other is not. Since the keys are identical either could be used to encrypt or decrypt data.

Each of the ICSF security controls is detailed below. These controls work together to implement Key Store policies. All are documented in the ICSF Administration Guide (SC14-7506). Note that these controls only apply to secure keys. All of these controls apply to the CKDS and PKDS, however the Key Archive Use control also applies to the TKDS.

Duplicate Key Token Checking

As described above, enabling this control instructs ICSF to look for an already existing copy of the key in the keystore, before inserting a new key. When a key record is created in the keystore, it includes a hash that represents the key material. As a new key is generated, a hash will be calculated for that key and if this security control is enabled, ICSF will search the keystore for a record already containing this hash value. If a matching hash already exists in the keystore, ICSF will not create the new record in the keystore.

This control can be enabled independently for the CKDS and PKDS. That is, duplicate key token checking can be enabled for the CKDS or the PKDS or both (or neither). To enable, simply create a RACF resource profile in the RACF XFACILIT class. The existence of these profiles tells ICSF to perform the check for a duplicate each time a keystore record is created.

Note that enabling this control will not identify duplicate records that are already defined in the keystore. ICSF provides a utility to look for duplicate records that already exist. This utility is invoked via a batch job referencing the keystore to be searched.

Key Token Authorization Checking

The ICSF APIs that require a key, support three different ways for the key to be passed. The most common technique is for the key label to be specified as a parameter to the API. Conveniently, the label is also used to provide security for the key material as profiles based on the label can be defined in the security product.

However, the key can also be passed to the API either as a raw key (the 8-, 16-, 24- or 32-byte key itself) or as a key token (a 64-byte structure as defined in the IBM Common Cryptographic Architecture) that also contains the key. In these two cases, there is no key label available to check the access authority and by default ICSF will not perform any authorization checks. When the APIs are invoked using either of these options, the API will execute without validating the user's authority to the key material.

Enabling the Key Token Authorization Checking control tells ICSF to perform a security check even if the label is not provided. In this case, ICSF will look for a label to use for the security check by searching the keystore for a record that contains the same key value. As described with the Duplicate Key Token Checking control, ICSF calculates a hash value of the key material and stores that hash with the key record. When no label is available to the API, ICSF will calculate a hash from the raw key or take the hash from the key token and then search the keystore for a record that contains that same hash value. If a match is found, ICSF will use that particular key label and look for a SAF profile to check the access list of the key. If a match is found, but the user is not authorized through that particular key label, ICSF will continue searching the keystore for another matching record. As soon as a match is found that permits access to the key, ICSF will allow the API to execute.

If all the records in the keystore are checked and no matching hash value is found, then the user is not specifically authorized to that key material. There is another Key Security control, Default Key Label Checking, described below that can provide a default security profile.

The Key Token Authorization checking control can be enabled in either WARN or FAIL mode. When using FAIL mode, if the user is not specifically authorized to use the key material, then the API will fail with a security violation. When using WARN mode, if the user is not authorized to use the key material, the API will be executed however warning messages and SMF records will be cut indicating that a user was allowed to use a particular key despite the lack of authorization. WARN mode would be used in a shop that has been using ICSF for some time but is just now enabling key authorization checks thus allowing production work to continue running while the new security controls are implemented and new access rules are put in place.

As with the Duplicate Key Token Checking control, this check can be enabled independently for the CKDS and PKDS, so it can be enabled for either, both or neither keystore.

Note: Activating either the Duplicate Key Token Checking or Key Token Authorization Checking control activates a key store policy for that particular keystore (CKDS or PKDS). This is a pre-req for some of the other controls.

Default Key Label Checking

The Default Key Label Checking control is closely related to the Key Token Authorization Checking control. As described above, if no label is found to perform the check, then RACF will respond that it cannot determine the access authority and ICSF will not allow the access. In other words, unless a key label is found and the access list for that key label includes the user trying to execute the API, then the operation will fail for a security violation.

The Default Key Label Checking control provides a default label to use when no other label is available. Once ICSF scans the keystore looking for a record that contains a hash that matches the key or key token being passed directly to the API, and does not find a match, the label specified in this default profile will be used to provide an access list.

Creating the Default Key Label Checking profile in RACF tells ICSF that if there were no labels to check the access list, then use this default profile. Specifically, for the CKDS, ICSF will look for the profile CSF-CKDS-DEFAULT and the profile CSF-PKDS-DEFAULT for the PKDS. ICSF will query RACF to see if the user has the appropriate access level in this default profile to determine whether the operation should proceed.

This default profile might be set up with UACC(READ) for a transition period when these controls are first implemented. That would allow any jobs that are using key material, without the property security in place, to continue running. However, enabling both the Key Token Authorization Checking control and the Default Key Label control, with UACC(NONE) for the default profile, has the added benefit of enforcing the requirement that all key material be defined in the keystore. By default, ICSF does not require that key material come from the keystore, however allowing key material to reside outside of the keystores introduces complexity into key management procedures and may not be as secure (i.e. how secure is the dataset where the key is stored?) By enabling both of these controls, ICSF will fail the use of any key that does not exist in the keystore, and thus does not have an access list that can be checked. Forcing all keys to be defined to ICSF and in the keystore provides for better key management and key security.

The Default Key Label control requires that the key store policy be activated for that particular keystore. As noted above, activating either the Key Token Authorization Check control or the Duplicate Key Token Check enables a key store policy that is a pre-req for the Default Key Label control. That is, this control has no affect unless one or both of those controls is also enabled.

Granular Key Label Access

Historically, ICSF access authority checks on key material either granted access or prohibited access no matter how the key was being used. That is, as long as the user had READ access to the key label, then they were able to perform whatever operation the API that they were using required. For a simple encrypt or decrypt operation, READ access provided sufficient authority to encrypt or decrypt data. However, if the API performed a create, update or even delete of a key, then READ access was also sufficient. This does not provide sufficient separation of duties for key material and beginning with HCR7770 ICSF implemented the Granular Key Label Access security control which strengthens the authorization levels required to support specific functions.

When this access control is enabled, ICSF will require a higher level of access authority for certain operations. READ authority is still sufficient for accessing the key for encrypt or decrypt operations. However, to create a new key with a new key label, the user must have UPDATE level authority to the key label. To update or delete an existing record in the keystore, the user must have CONTROL authority for that key label.

As an example, when an API such as Key Record Delete is executed, if the Granular Key Label Access control is enabled, the user must have CONTROL authority for the particular label. Otherwise the API will fail with an access violation.

This control can also be enabled in either WARN or FAIL mode. When FAIL mode is used, the operation will fail if the user does not have a sufficient access level to the key label. When WARN mode is used, the operation will complete successfully, in spite of the user not having sufficient authority. However, ICSF will generate a message in the job log indicating that a resource was accessed without sufficient authority. ICSF will also cut an SMF record to record the event.

This control does not require an active keystore policy, however unless the Key Token Authorization Check and Duplicate Key Token controls are in place, then a key token or just the key value could be passed to the API and there would be no label to use for the security profile to be checked.

Symmetric Key Label Export

The Symmetric Key Label Export control, like the Granular Key Label Access control, provides more granularity to manage access to key material, but specifically for protecting keys involved in export operations.

The Symmetric Key Export API will take a secure symmetric key, either AES or DES, from the CKDS and wrap (encrypt) it using a public key. Typically, this API would be used when a symmetric key needs to be sent, securely, to a partner, and the public key that protects the symmetric key would have come from that partner, probably via digital certificate. So the key would need to be accessed for performing an encryption operation and it would also need to be accessed for wrapping and sending to a partner. Separation of duties would require that the user who performs the encrypt operation should not be permitted to extract and wrap the key for the partner. Similarly, the key officer that is responsible for extracting and wrapping the key should not have authority to use that key in encrypt or decrypt operations.

Without the Symmetric Key Label Export control, the API will simply look for READ access in the CSFKEYS class profile for the symmetric key that will be exported. Enabling the Symmetric Key Label Export control changes the security checks for authority to access the symmetric key. When this control is enabled, the Symmetric Key Export API will instead look for UPDATE authority in the XCSFKEY class. The CSFKEYS class profile is still required, because it will be used by the encrypt/decrypt APIs to verify that the user is authorized to access the key. However, for export operations, the XCSFKEY class is now used to validate authority to the symmetric key. The export API will also still check the CSFKEYS class for READ access to the RSA public key which is used to wrap the symmetric key.

Enabling this control requires a couple of steps. Logically, the first step is to create the XCSFKEY profile with appropriate access lists for the keys that will be exported. After that is created, the control must be enabled. (If the control is enabled before the XCSFKEY profile is created, the API may fail because the user is not in the access list.) The control can be enabled independently for AES and/or DES keys. Creating the CSF.XCSFKEY.ENABLE.AES tells ICSF to perform the additional checks on AES keys and creating the CSF.XCSFKEY.ENABLE.DES tells ICSF to perform the additional checks on DES keys.

Like the Granular Key Label Access control, the Symmetric Key Label Export control does not require a keystore policy to be in place, however, without it, there is the possibility that a key passed to the API as a key token or raw key could be exported without checking access authority.

PKA Key Management Extensions

The PKA Key Management Extensions control provides additional checks on whether a symmetric key can be exported at all and which PKA keys can be used for the export. It can also be used to control whether an asymmetric key can be used in secure export and import operations or in handshake operations, or both.

Public/Private Key architecture provides for authentication (using handshakes) as well as key management (securely importing and exporting key material). By default, ICSF does not differentiate between those PKA operations, however good crypto practice does require restrictions. In general, a key that is used for authenticating a party should not also be used for exporting and importing symmetric keys. And keys that are used for wrapping symmetric keys should not be used for authentication. Enabling the PKA Key Management Extension control tells ICSF to perform these additional checks. So a public/private key pair can be limited to authentication or import/export, but not both.

Enabling this control tells ICSF to look at the ICSF segment in the CSFKEYS or XCSFKEY class profile for additional information about key usage. The ICSF segment can contain a field called ASYMUSAGE (for Asymmetric usage) and that field can contain the value SECUREEXPORT or NOSECUREEXPORT, indicating whether that particular key can be used to export symmetric keys. The ASYMUSAGE field can also contain HANDSHAKE or NOHANDSHAKE, indicating whether it can be used for performing authentication operations.

By default, an RSA key can be used in secure import and export operations. If the ASYMUSAGE is not specified or if it is explicitly set to SECUREEXPORT, then the RSA key can be used for secure key import/export operations. If the ASYMUSAGE field is set to NOSECUREEXPORT, then that key cannot be used for secure import/export operations.

Similarly, by default, a PKA key can also be used in handshake operations. If the ASYMUSAGE is not specified or if it is explicitly set to HANDSHAKE then the PKA key can be used for authentication. If the ASYMUSAGE field is set to NOHANDSHAKE, then that PKA key cannot be used for authentication.

Enabling the PKA Key Management Extension control also provides some additional security for the symmetric key in an export operation. Enabling this control can also check that a symmetric key, covered by a CSFKEYS or XCSFKEY profile:

- Cannot be exported
- Can be exported by any asymmetric key in the PKDS
- Can be exported, but only using asymmetric keys from a supplied list
- Can be exported by an asymmetric key that is bound to an identity in a certificate from a trusted certificate repository (either a PKCS #11 token or a SAF key ring)
- Can be exported only by an asymmetric key that is bound to specific identities as defined in a list of key certificates in a trusted certificate repository)

These controls for the symmetric key are also implemented in the ICSF segment by specifying the SYMEXPORTABLE field. The SYMEXPORTABLE field can contain:

- BYNONE – the symmetric cannot be exported
- BYLIST – two other ICSF segment fields, SYMEXPORTKEYS or SYMEXPORTCERTS provide a list of keys or certificates that can be used to identify keys that can wrap the symmetric key
- BYANY – the symmetric can be exported (as long as there are no other restrictions in place)

When the BYLIST option is specified for the SYMEXPORTABLE field, then ICSF will look for the label of the public key passed to the API in the list of keys in SYMEXPORTKEYS. If found, the operation will proceed, (assuming the user has authority to use that public key). Alternatively, instead of supplying the label of a public key, the label of a certificate can be provided. When SYMEXPORTCERTS is specified in the ICSF segment, the API will use the public key that is bound to the certificate to protect the symmetric key.

The PKA Key Management Extensions control can be enabled either in WARN or FAIL mode. As with other controls, if FAIL mode is specified, then the operation will fail unless the appropriate authorities are in place. With WARN mode, the operation will succeed, but messages will be issued to the job log and SMF records cut indicating that the proper authorities were not in place. Warn mode is intended for environments where the appropriate definitions are being implemented for the first time.

The PKA Key Management Extensions control requires that the Key Store policy be activated for that particular keystore. As described above, activating the Key Store Policy requires that either the Key Token Authorization Checking control or the Duplicate Key Token Check, or both, be enabled.

Key Archive Use

This newest control is associated with the new Archive flag that was introduced with HCR77B0. That version of ICSF introduced new date fields including a date last used field. That date can then be evaluated and if enough time has passed the record can be marked as 'ARCHIVED'. That provides the ability to retain the key in the keystore but prohibit it from being used. By default, once a key has it's archive flag turned on, ICSF will not allow it to be used.

To ease the implementation of procedures for identifying archived keys, ICSF does provide a way to override this default function. Creating the CSF.KDS.ARCHIVE.USE profile in the XFACILIT class tells ICSF to permit the use of a key even though the archive flag is on. ICSF will generate messages and SMF records indicating that a particular key was used, even though the archive flag was on, but it will allow the job to access the key.

Creating the profile enables the use of archived keys in both the CKDS and PKDS. Deleting the profile restores ICSF to its default operation of prohibiting the use of archived keys.

Recommendations

Each of these controls provides additional security for key material, over and above simple access authority. A shop just implementing crypto on z/OS should consider implementing each of these controls, except Key Archive Use from the beginning. For example, enabling the Duplicate Key Token Check before any keys are defined means that you don't have to check for duplicate keys that might already exist in the keystore.

The Key Archive Use control allows archived keys to be used while implementing and testing key archive policies. A new installation will not be archiving keys and doesn't have to worry about a badly implemented policy.

For other controls, a shop that is already using the crypto infrastructure extensively and has many keys defined in the keystores needs to proceed slowly. Using WARN mode allows the policies to be implemented without preventing production workloads from running.

New ICSF: HCR77C0

In case you didn't notice, IBM announced a new release of ICSF. On October 5, in announcement letter 216-392, IBM announced HCR77C0, ''Cryptographic Support for z/OS V2R1 – z/OS V2R2".

The bulk of the enhancements are related to enhancing existing APIs or adding new algorithm support, such as RSA-PSS Signatures. However, there is also a new facility to refresh the Options Data Set. It sounds like you can pick up changes to the options data set without shutting down and restarting ICSF. There is also a bullet in the announcement letter about 'Improved Key Lifecycle and Key Usage auditing'.

This web deliverable became available for download on October 17 from the z/OS Downloads website, http://www.ibm.com/systems/z/os/zos/tools/downloads.

Look for a more in-depth look at this new release in a future newsletter.

Mainframe Crypto: How can we be of service?

Mainframe Crypto is happy to help with any aspect of implementing crypto on z Systems. Whether it would be getting the infrastructure in place or rolling out a specific implementation like encrypting databases or securing network communications we'd be happy to assist.

We still see a lot of shops that have implemented crypto, but don't really understand all of the capabilities. If you rolled out crypto to protect your backups then you're only scratching the surface of the protection that is available. We'd like the opportunity to help you evaluate your environment and look for opportunities to more fully utilize the breadth of crypto.

No matter whether you're just getting started with crypto on System z or you've got an established crypto environment, Mainframe Crypto can help!

NewEra zExchange

NewEra Software continues to offer the zExchange, with presentations on lots of topics on z Systems.

My next crypto session is scheduled for Nov. 30 at 1PM. Tentatively it will be on database encryption and some new options for securing your DB2 and IMS databases.

You can see the monthly schedule and register at http://www.newera-info.com/Month.html.

[illegible]