Security Enhancements for HSTS 3.9.6.2

IBM Aspera HSTS/HSTE now provides a facility for securely managing secrets and keys that pose a security risk when stored on machines in plain-text format. To address these risks, HSTS/HSTE includes a new tool, **`askmscli`**, that provides the following capabilities:

- **Content protection**: Encrypting passphrases used for server-side encryption at rest (SSEAR).
- **Dynamic token encryption key**: The key used for encrypting authorization tokens can be dynamically generated for improved security and time-limited validity.
- **Master key for token encryption keys**: The dynamic token encryption key used for encrypting authorization tokens can be encrypted using a master key.
- **Master key for Redis database**: Encryption of sensitive information in a Redis database, using a secure 256-bit master key set by the system administrator. Sensitive information, for example, could be an access-key-specific content-protection secret or token encryption key.

Migration Procedures

The migration steps provided in this topic allow you to take full advantage of the new security features. Although not required, Aspera strongly recommends you adopt these measures to ensure the secure usage of Aspera products. The steps consist of running the **`askmscli`** tool, which stores secrets in file-system-protected keystore files rather than in the `aspera.conf` file.

According to which encryption features you're setting up, the **`askmscli`** command recognizes three categories of secrets:

`ssear`	Passphrases used for server-side encryption at rest (SSEAR).
`redis-master-key`	Master key for encryption of sensitive data in a Redis database, such as token encryption keys.

The **`askmscli`** tool stores secrets in SQLite DBs. It maintains the encrypted secrets in two databases:

`rootkeystore.db`	Functions as a backup and main source of truth for encrypted secrets.
`localkeystore.db`	Resides in user's home directory (by default). It contains a copy of shared keys like `redis-master-key` along with a user's own secrets. This allows every user to use the same shared keys without requiring a shared/world-readable database.

NOTE: The procedures in the sections below are organized to indicate which steps are for upgrades, which are for new installations, and which are for both.

The commands in this section are shown with a standard Linux/macOS shell prompt ($) and the commands are identical for both OSs. On Windows, the syntax for **`askmscli.exe`** is identical. All Windows commands must be run with Administrator permissions (and without **sudo**). In cases where a Windows command invocation is different, the Windows invocation is shown separately and identified as such.

On Linux systems, all commands in this topic are located in `/opt/aspera/bin`. For convenience, you may want to add `/opt/aspera/bin` to your path.

On macOS, the these commands are found in `/Library/Aspera/bin` and `/Library/Aspera/sbin`. For convenience, you may want to add these to your path.

On Windows, these commands are located in `C:\Program Files\Aspera\Enterprise Server\bin`
.

Content-Protection Secret

The **askmscli** tool sets content-protection secrets only for each user, not for groups and not for all users on a node. Each transfer user now requires their own content-protection secret for SSEAR. The steps below describe how to migrate content-protection secrets from `aspera.conf` to a user's local keystore. Two procedures are shown: one for new HSTS/HSTE installations and one for upgrades. Aspera recommends that you no longer store content-protection secrets in `aspera.conf`.

For Upgrades:

Before you start, make a backup copy of your `aspera.conf` file.

1. Locate all `content_protection_secret` settings in `aspera.conf`, and make a note of the value that's set for each.
2. Set a content-protection secret for each user:

   ```
   $ echo -n secret | sudo askmscli -u username -s ssear
   ```

   Windows (as Administrator):

   ```
   > echo secret | askmscli.exe -u username -s ssear
   ```

3. Remove all plain-text content-protection secrets from `aspera.conf`:

   ```
   $ sudo asconfigurator -x "set_user_data; user_name,user;
    transfer_encryption_content_protection_secret,AS_NULL"
   $ sudo asconfigurator -x "set_group_data; group_name,group;
    transfer_encryption_content_protection_secret,AS_NULL"
   $ sudo asconfigurator -x "set_node_data;
    transfer_encryption_content_protection_secret,AS_NULL"
   ```

For New Installations:

1. Set the content-protection secret for each transfer user by running this command:

   ```
   $ echo -n secret | sudo askmscli -u username -s ssear
   ```

   Windows (as Administrator):

   ```
   > echo secret | askmscli.exe -u username -s ssear
   ```

Token Encryption Key

The following is a brief summary of the steps for encrypting and using dynamic token encryption keys:

(1) Remove existing plain-text token encryption keys from `aspera.conf`.
(2) Set `token_dynamic_key` to `true` in `aspera.conf`.
(3) Set a master key for Redis.

Before you proceed, make a backup copy of your `aspera.conf` file.

For Upgrades and New Installations:

1. Enable the use of dynamic token encryption keys by setting `token_dynamic_key` to `true` in `aspera.conf`.

   ```
   $ sudo asconfigurator -x "set_node_data; token_dynamic_key,true"
   ```

   NOTE: A dynamic token encryption key can be set for an individual user or a system group.

2. Set a Redis master key using **askmscli**. The master key must be a unique random 256-bit key. The example below uses **openssl** to generate the key. This Redis master key will be used to encrypt the dynamic token encryption key.

   ```
   $ echo -n "`openssl rand -base64 32`" | sudo askmscli -s redis-master-key
   ```

   Windows (as Administrator):

   ```
   > openssl rand -base64 32 > redis_master_key_file
   > type redis_master_key_file | askmscli.exe -s redis-master-key
   ```

3. For each transfer user with a token encryption key, run the commands below to initialize the user's keystore:

   ```
   $ sudo askmscli -i -u username
   ```

   Windows (as Administrator):

   ```
   > askmscli.exe -u username -i -L- -DD
   ```

   Using this command, also initialize the keystore for the user `asperadaemon` that runs **asperanoded**.

4. Restart **asperanoded** to apply the new configuration changes. To test transfers, try an upload and download through your Web application.

   For Upgrades Only:

5. Once all the outstanding tokens created from the old token encryption keys have expired, remove the `token_encryption_key` settings from `aspera.conf`:

   ```
   $ sudo asconfigurator -x "set_user_data; user_name,user; token_encryption_key,AS_NULL"
   $ sudo asconfigurator -x "set_group_data; group_name,group; token_encryption_key,AS_NULL"
   $ sudo asconfigurator -x "set_node_data; token_encryption_key,AS_NULL"
   ```

Redis Master Key

System administrators can now set a unique 256-bit Redis master key to encrypt local access-key configuration and dynamic token encryption keys.

For New Installations Only: (Not applicable to upgrades.)

If you have not already created a Redis master key as part of enabling dynamic token encryption keys (as in "Token Encryption Key" above), then:

1. Set a Redis master key using **askmscli**. The master key must be a unique, random 256-bit key. The example below uses **openssl** to generate the key. This Redis master key will be used to encrypt both the dynamic token encryption key and access keys in Redis.

   ```
   $ echo -n "`openssl rand -base64 32`" | sudo askmscli -s redis-master-key
   ```

   Windows (as Administrator):

   ```
   > openssl rand -base64 32 > redis_master_key_file
   > type redis_master_key_file | askmscli.exe -s redis-master-key
   ```

2. For the transfer user associated with the an access key, run **askmscli** to initialize the user's keystore (if not done previously):

   ```
   $ sudo askmscli -i -u username
   ```

   Windows (as Administrator):

   ```
   > askmscli.exe -u username -i -L- -DD
   ```

   Using this command, also initialize the keystore for the user `asperadaemon` that runs **asperanoded** (if not done previously)

3. Encrypt your access key data by running the **asnodeadmin** command:

```
$ sudo asnodeadmin --encrypt-access-key --access-key access_key
```