NVDIMM Security

1. Introduction

With the introduction of Intel Device Specific Methods (DSM) v1.8 specification [1], security DSMs are introduced. The spec added the following security DSMs: "get security state", "set passphrase", "disable passphrase", "unlock unit", "freeze lock", "secure erase", and "overwrite". A security_ops data structure has been added to struct dimm in order to support the security operations and generic APIs are exposed to allow vendor neutral operations.

2. Sysfs Interface

The "security" sysfs attribute is provided in the nvdimm sysfs directory. For example: /sys/devices/LNXSYSTM:00/LNXSYBUS:00/ACPI0012:00/ndbus0/nmem0/security

The "show" attribute of that attribute will display the security state for that DIMM. The following states are available: disabled, unlocked, locked, frozen, and overwrite. If security is not supported, the sysfs attribute will not be visible.

The "store" attribute takes several commands when it is being written to in order to support some of the security functionalities: update <old_keyid> <new_keyid> - enable or update passphrase. disable <keyid> - disable enabled security and remove key. freeze - freeze changing of security states. erase <keyid> - delete existing user encryption key. overwrite <keyid> - wipe the entire nvdimm. master_update <keyid> <new_keyid> - enable or update master passphrase. master_erase <keyid> - delete existing user encryption key.

3. Key Management

The key is associated to the payload by the DIMM id. For example: # cat /sys/devices/LNXSYSTM:00/LNXSYBUS:00/ACPI0012:00/ndbus0/nmem0/nfit/id 8089-a2-1740-00000133 The DIMM id would be provided along with the key payload (passphrase) to the kernel.

The security keys are managed on the basis of a single key per DIMM. The key "passphrase" is expected to be 32bytes long. This is similar to the ATA security specification [2]. A key is initially acquired via the request_key() kernel API call during nvdimm unlock. It is up to the user to make sure that all the keys are in the kernel user keyring for unlock.

A nvdimm encrypted-key of format enc32 has the description format of: nvdimm:<bus-provider-specific-unique-id>

See file `Documentation/security/keys/trusted-encrypted.rst` for creating encrypted-keys of enc32 format. TPM usage with a master trusted key is preferred for sealing the encrypted-keys.

4. Unlocking

When the DIMMs are being enumerated by the kernel, the kernel will attempt to retrieve the key from the kernel user keyring. This is the only time a locked DIMM can be unlocked. Once unlocked, the DIMM will remain unlocked until reboot. Typically an entity (i.e. shell script) will inject all the relevant encrypted-keys into the kernel user keyring during the initramfs phase. This provides the unlock function access to all the related keys that contain the passphrase for the respective nvdimms. It is also recommended that the keys are injected before libnvdimm is loaded by modprobe.

5. Update

When doing an update, it is expected that the existing key is removed from the kernel user keyring and reinjected as different (old) key. It's irrelevant what the key description is for the old key since we are only interested in the keyid when doing the update operation. It is also expected that the new key is injected with the description format described from earlier in this document. The update command written to the sysfs attribute will be with the format: update <old keyid> <new keyid>

If there is no old keyid due to a security enabling, then a 0 should be passed in.

6. Freeze

The freeze operation does not require any keys. The security config can be frozen by a user with root privelege.

7. Disable

The security disable command format is: disable <keyid>

An key with the current passphrase payload that is tied to the nvdimm should be in the kernel user keyring.

8. Secure Erase

The command format for doing a secure erase is: erase <keyid>

An key with the current passphrase payload that is tied to the nvdimm should be in the kernel user keyring.

9. Overwrite

The command format for doing an overwrite is: overwrite <keyid>

Overwrite can be done without a key if security is not enabled. A key serial of 0 can be passed in to indicate no key.

The sysfs attribute "security" can be polled to wait on overwrite completion. Overwrite can last tens of minutes or more depending on nvdimm size.

An encrypted-key with the current user passphrase that is tied to the nvdimm should be injected and its keyid should be passed in via sysfs.

10. Master Update

The command format for doing a master update is: update <old keyid> <new keyid>

The operating mechanism for master update is identical to update except the master passphrase key is passed to the kernel. The master passphrase key is just another encrypted-key.

This command is only available when security is disabled.

11. Master Erase

The command format for doing a master erase is: master_erase <current keyid>

This command has the same operating mechanism as erase except the master passphrase key is passed to the kernel. The master passphrase key is just another encrypted-key.

This command is only available when the master security is enabled, indicated by the extended security status.

[1]: https://pmem.io/documents/NVDIMM_DSM_Interface-V1.8.pdf

[2]: http://www.t13.org/documents/UploadedDocuments/docs2006/e05179r4-ACS-SecurityClarifications.pdf