Solid state drive/Memory cell clearing

< Solid state drive

Related articles
Solid State Drive
Securely wipe disk

On occasion, users may wish to completely reset the SSD to the initial "clean" state it was manufactured with, thus restoring it to its **factory default write performance (https://www.anandtech.com/storage/showdoc.aspx?i=3531&p=8)**. Write performance is known to degrade over time even on SSDs with native TRIM support. TRIM only safeguards against file deletes, not replacements such as an incremental save.

Performing the Secure Erase does not reset the wear leveling status of SSD cells - a drive close to the end of its lifespan *may* become writable for a short while, but it will still fail after a limited amount of writes.

Warning:

- Back up all data of importance prior to continuing! Using **this procedure will destroy all data on the SSD** and render it unrecoverable by even data recovery services! Users will have to repartition the device and restore the data after completing this procedure!
- Do **not** proceed with this if the target drive is not connected directly to a SATA/NVMe interface. Issuing the Secure Erase/Format/Sanitize command on a drive connected via USB or a SAS/RAID card could potentially brick the drive!

1 SATA drive

ATA has two commands for wiping a drive— `SECURITY ERASE UNIT` and `ENHANCED SECURITY ERASE UNIT`.**[1] (https://ata.wiki.kernel.org/index.php/ATA_Secure_Erase)**

Tip: These commands can also be used to zero out hard disk drives.

1.1 Make sure the drive security is not in frozen mode

Issue the following command:

```
# hdparm -I /dev/sdX | grep frozen
```

In the security section of the output it should say `not frozen`. If it shows as just `frozen` then you cannot continue to the next step. See **Solid state drive#Frozen mode** for details.

A possible solution is to simply **suspend** (using S3 not S0ix) the system. Upon waking up, it is likely that the freeze will be lifted. If unsuccessful, one can try hot-(re)plug the data cable (which might crash the kernel). If hot-(re)plugging the SATA data cable crashes the kernel try letting the operating system fully boot up, then quickly hot-(re)plug both the SATA power and data cables. If hot-(re)plugging of SATA cables still crashes the kernel, make sure that AHCI is enabled in the BIOS

(AHCI allows hot-(re)plugging operations without a crash). Using a USB-to-SATA adapter is an option if it supports hotplugging. One can also use **hdparm (https://archlinux.org/packages/?name=hdparm)** via USB.

1.1.1 Dell Systems

If the command output shows "frozen", you may be able to work around it by:

1. Reboot into the Dell BIOS by pressing F2 on startup.
2. Set the Internal HDD Password in the BIOS (be careful, the keymap is en_US / qwerty).
3. Apply the changes and reboot.
4. When prompted for the password by Dell Security Manager, press Escape rather than entering it. The drive will remain locked but not frozen.
5. Skip step 2, and go directly to Step 3 below.

Note: If you are using a Lenovo system and can not remove the "frozen" state (e.g. Lenovo tablets use SSD on M.2 interface), you can use a **proprietary tool (https://pcsupport.lenovo.com/us/en/olddownloads/ds019026)** to accomplish the memory cell clearing rather than following this article. See also: **https://superuser.com/questions/763642/secure-erase-ssd-on-lenovo-thinkpad-t520-cant-unfreeze-ssd-machine-reboots-on**

1.2 Enable security by setting a user password

Note: When the user password is set the drive will be locked after next power cycle denying normal access until unlocked with the correct password.

Warning: Do **not** reboot your computer after this step, particularly if you have a Lenovo laptop. Certain variants of Lenovo's BIOS are susceptible to use a deviating algorithm for calculating the encryption key. After startup the machine will not be able to connect the SSD drive.**[2] (https://jbeekman.nl/blog/2015/03/lenovo-thinkpad-hdd-password/)**

Any password will do, as this should only be temporary. After the secure erase the password will be set back to NULL. In this example, the password is `PasSWorD` as shown:

```
# hdparm --user-master u --security-set-pass PasSWorD /dev/sdX
```

```
security_password="PasSWorD"
/dev/sdX:
Issuing SECURITY_SET_PASS command, password="PasSWorD", user=user, mode=high
```

As a sanity check, issue the following command

```
# hdparm -I /dev/sdX
```

The command output should display "enabled":

```
Security:
        Master password revision code = 65534
                supported
                enabled
        not     locked
        not     frozen
        not     expired: security count
                supported: enhanced erase
```

```
        Security level high
        2min for SECURITY ERASE UNIT. 2min for ENHANCED SECURITY ERASE UNIT.
```

1.3 Issue the ATA SECURITY ERASE UNIT command

The final step is to issue the ATA SECURITY ERASE UNIT command, instructing the device's firmware to erase its contents. Note for the device used in this example, earlier output states:

```
2min for SECURITY ERASE UNIT. 2min for ENHANCED SECURITY ERASE UNIT.
```

As per ATA specification the *enhanced* security erase (`--security-erase-enhanced`) performs a more elaborate wipe. If the estimated completion time for both commands is equal, it indicates the drive manufacturer shortcut the specification and uses the same erase function for both. A short time (like 2 minutes) in turn indicates the device is self-encrypting and its firmware function will wipe the internal encryption key instead of overwriting all data cells.**[3] (https://security.stackexchange.com/questions/62253/what-is-the-difference-between-ata-secure-erase-and-security-erase-how-can-i-en)**

Warning:

- Triple check that the correct drive designation is used. There is **no turning back** once the command is confirmed. You have been warned.
- Ensure that the drive is not mounted when this is ran. If a secure erase command is issued while the device is mounted, it will not erase properly.

```
# hdparm --user-master u --security-erase PasSWorD /dev/sdX
```

Wait until the command completes. This example output shows it took about 40 seconds for an Intel X25-M 80GB SSD.

```
security_password="PasSWorD"
/dev/sdX:
Issuing SECURITY_ERASE command, password="PasSWorD", user=user
0.000u 0.000s 0:39.71 0.0%      0+0k 0+0io 0pf+0w
```

The drive is now erased. After a successful erasure the drive security should automatically be set to disabled (thus no longer requiring a password for access). Verify this by running the following command:

```
# hdparm -I /dev/sdX
```

The command output should display "not enabled":

```
Security:
        Master password revision code = 65534
                supported
        not     enabled
        not     locked
        not     frozen
        not     expired: security count
                supported: enhanced erase
        2min for SECURITY ERASE UNIT. 2min for ENHANCED SECURITY ERASE UNIT.
```

2 NVMe drive

The **NVMe specification (https://nvmexpress.org/developers/nvme-specification/)** defines a standardized way to format NVMe drives, since those do not use the SATA interface protocol and therefore cannot be cleared in the same way as SATA SSDs. Originally it was the **`nvme-format(1)` (https://man.archlinux.org/man/nvme-format.1)** command (part of the **`nvme-cli` (https://archlinux.org/packages/?name=nvme-cli)**) which provided this feature, but while it still does **Specification 1.3 (https://nvmexpress.org/changes-in-nvme-revision-1-3/)** added support for a dedicated **`nvme-sanitize(1)` (https://man.archlinux.org/man/nvme-sanitize.1)** command. As **described by the NVM Express Consortium (https://nvmexpress.org/open-source-nvme-management-utility-nvme-command-line-interface-nvme-cli/)**:

> These commands are used to securely erase user data from the device. This can be used when deploying a new device, retiring or at device end-of-life, using an SSD for a new application and so on. Sanitize was introduced in NVMe 1.3 specification, so before then NVMe Format was used exclusively to perform secure erase. While both options work, Sanitize is more robust for ensuring the data was properly wiped; format is good for everyday use and testing.

In order to verify what is supported by your drive, use the Identify Controller command:

```
# nvme id-ctrl /dev/nvme0 -H | grep -E 'Format |Crypto Erase|Sanitize'
```

Example output:

```
  [1:1] : 0x1   Format NVM Supported
  [29:29] : 0   No-Deallocate After Sanitize bit in Sanitize command Supported
    [2:2] : 0   Overwrite Sanitize Operation Not Supported
    [1:1] : 0x1 Block Erase Sanitize Operation Supported
    [0:0] : 0x1 Crypto Erase Sanitize Operation Supported
  [2:2] : 0x1   Crypto Erase Supported as part of Secure Erase
  [1:1] : 0     Crypto Erase Applies to Single Namespace(s)
  [0:0] : 0     Format Applies to Single Namespace(s)
```

Then proceed with either **format** or **sanitize**.

2.1 Format command

The Format command is conceptually closer to a mix of **hdparm** and **fdisk**, as it allows to set low-level parameters for the drive *and* additionally to send a secure erase command.

`nvme-format(1)` (https://man.archlinux.org/man/nvme-format.1) gives the following details about the Secure Erase Settings (`-s` / `--ses`) option:

> Secure Erase Settings: This field specifies whether a secure erase should be performed as part of the format and the type of the secure erase operation. The erase applies to all user data, regardless of location (e.g., within an exposed LBA, within a cache, within deallocated LBAs, etc). Defaults to 0.

Possible values :

Value	Definitions
0	No secure erase operation requested
1	User Data Erase: All user data shall be erased, contents of the user data after the erase is indeterminate (e.g., the user data may be zero filled, one filled, etc). The controller may perform a cryptographic erase when a User Data Erase is requested if all user data is encrypted.
2	Cryptographic Erase: All user data shall be erased cryptographically. This is accomplished by deleting the encryption key.

While the Format command accepts either the whole NVMe character device (e.g. `/dev/nvme0`) or a specific namespace block device (e.g. `/dev/nvme0n1`), make sure this feature is supported by your drive before triggering it. E.g. in the output of the Identify Controller command above, we see that the `Crypto Erase Applies to Single Namespace(s)` and `Format Applies to Single Namespace(s)` bits are set to zero, which according to the spec means that "the controller supports format on a per namespace basis" (see figure 249 byte row 524 "Format NVM Attributes (FNA)").

For example, to format `/dev/nvme0` with a crypto erase to namespace 1:

```
# nvme format /dev/nvme0 -s 2 -n 1
```

Use `-n 0xffffffff` to format all the namespaces.

See **`nvme-format(1)` (https://man.archlinux.org/man/nvme-format.1)** for more information and important warnings regarding device/namespace selection.

2.2 Sanitize command

The Sanitize command was created to be "functionally equivalent to the command of the same name in SATA and SAS implementations"**[4] (https://nvmexpress.org/changes-in-nvme-revision-1-3/)**. From the **aforementioned article (https://web.archive.org/web/20200430084547/https://nvmexpress.org/open-source-nvme-management-utility-nvme-command-line-interface-nvme-cli/)**:

> According to the **NVMe 1.4 specification (https://nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf#%5B%7B%22num%22%3A1051%2C%22gen%22%3A0%7D%2C%7B%22name%22%3A%22XYZ%22%7D%2C70%2C369%2C0%5D)**, "a sanitize operation alters all user data in the NVM subsystem such that recovery of any previous user data from any cache, the non-volatile media, or any Controller Memory Buffer is not possible."
> The big difference between Sanitize and Format is that sanitize ensures caches are deleted, and the process starts again after an unexpected power loss. Sanitize also supports a pattern overwrite for a secure erase operation, which is terrible for NAND endurance but can be used with other types of storage and memory classes, or for more certainty that user data cannot be recovered.

Usage and possible values for the `-a/--sanact` options are described in **`nvme-sanitize(1)` (https://man.archlinux.org/man/nvme-sanitize.1)**.

The difference between Block Erase and Crypto Erase is that Crypto Erase only erases an encryption key (as defined in the **NVMe 1.4 specification (https://nvmexpress.org/wp-content/uploads/NVM-Express-1_4-2019.06.10-Ratified.pdf#%5B%7B%22num%22%3A663%2C%22gen%22%3A0%7D%2C%7B%22name%22%3A%22XYZ%22%7D%2C70%2C567%2C0%5D)**):

> User Data Erase: All user data shall be erased, contents of the user data after the erase is indeterminate (e.g., the user data may be zero filled, one filled, etc.). The controller may perform a cryptographic erase when a User Data Erase is requested if all user data is encrypted.
> Cryptographic Erase: All user data shall be erased cryptographically. This is accomplished by deleting the encryption key.
> …
> The Block Erase sanitize operation alters user data with a low-level block erase method that is specific to the media for all locations on the media within the NVM subsystem in which user data may be stored;
> The Crypto Erase sanitize operation alters user data by changing the media encryption keys for all locations on the media within the NVM subsystem in which user data may be stored…

You can get an estimation of the time the various methods would take on your drive (if supported):

```
# nvme sanitize-log /dev/nvme0
```

```
...
Estimated Time For Overwrite                  :  4294967295 (No time period reported)
Estimated Time For Block Erase                :  174
Estimated Time For Crypto Erase               :  34
```

If instead you get a result such as:

```
# nvme sanitize-log /dev/nvme0
```

```
...
Estimated Time For Overwrite                  :  4294967295 (No time period reported)
Estimated Time For Block Erase                :  4294967295 (No time period reported)
Estimated Time For Crypto Erase               :  4294967295 (No time period reported)
```

Then be sure that the operation **will** take a long time to complete. For reference, a Block Erase took around 2-3 hours to complete on the Intel 660p 512GB reporting those results.

Warning: Once started, this operation is uninterruptible—even with a power cycle—and will render the drive unusable until the process completes, which can take **a long time (https://github.com/linux-nvme/nvme-cli/issues/752#issuecomment-647821283)**.

To start a Crypto Erase sanitize operation:

```
# nvme sanitize device -a start-crypto-erase
```

Note: The `device` parameter is mandatory NVMe character device (e.g. `/dev/nvme0` and **not** `/dev/nvme0n1`), as the operation applies necessarily to **whole devices**.

For a Block Erase:

```
# nvme sanitize device -a start-block-erase
```

Warning: Avoid using the *Overwrite* action even if it is "**not good or recommended for NAND based SSDs due to endurance (https://nvmedevelopers.com/English/Collaterals/Proceedings/2018/20181204_PRECON2_Hands.pdf)**[**dead link** 2024-01-13 ⓘ]".

You can follow the progress with the Sanitize Log:

```
# nvme sanitize-log /dev/nvme0
```

Example output for a drive with a Crypto Erase in progress:

```
Sanitize Progress                      (SPROG) :  655
Sanitize Status                        (SSTAT) :  0x4
Sanitize Command Dword 10 Information (SCDW10) :  0x4
Estimated Time For Overwrite                   :  4294967295 (No time period reported)
Estimated Time For Block Erase                 :  174
Estimated Time For Crypto Erase                :  34
Estimated Time For Overwrite (No-Deallocate)   :  0
Estimated Time For Block Erase (No-Deallocate) :  0
Estimated Time For Crypto Erase (No-Deallocate):  0
```

When the command has completed successfully:

```
Sanitize Progress                      (SPROG) :  65535
Sanitize Status                        (SSTAT) :  0x101
Sanitize Command Dword 10 Information (SCDW10) :  0x4
Estimated Time For Overwrite                   :  4294967295 (No time period reported)
Estimated Time For Block Erase                 :  174
Estimated Time For Crypto Erase                :  34
Estimated Time For Overwrite (No-Deallocate)   :  0
Estimated Time For Block Erase (No-Deallocate) :  0
Estimated Time For Crypto Erase (No-Deallocate):  0
```

3 Common method with blkdiscard

The **blkdiscard(8) (https://man.archlinux.org/man/blkdiscard.8)** command from the **util-linux (https://archlinux.org/packages/?name=util-linux)** package offers a `--secure` option to "Perform a secure discard. A secure discard is the same as a regular discard except that all copies of the discarded blocks that were possibly created by garbage collection must also be erased. This requires support from the device".

To use it, execute:

```
# blkdiscard --secure /dev/device
```

For devices which do not support a secure erase, a `-z` / `--zeroout` option fills the device with zeroes instead of simply discarding all blocks on the device by default.

See **[5] (https://unix.stackexchange.com/questions/659931/how-secure-is-blkdiscard)** for a discussion of the general security of *blkdiscard* and **[6] (https://www.ovirt.org/develop/release-management/features/storage/wipe-volumes-using-blkdiscard.html)** for an illustration of wiping volumes using blkdiscard.

4 Troubleshooting

4.1 UEFI boot entries get removed after wiping a drive

Some **UEFI** implementations remove boot entries referencing nonexistent files upon system startup. If you plan to restore the system from a backup after memory cell clearing, make sure to also restore the boot entry using **efibootmgr** or by reinstalling the **boot loader**.

5 See also

- **Secure Erase HDDs/SSDs (SATA/NVMe) using hdparm & nvme-cli on Linux (https://web.archive.org/web/20190326142043/http://forum.notebookreview.com/threads/secure-erase-hdds-ssds-sata-nvme-using-hdparm-nvme-cli-on-linux.827525/)** (2019): good tutorial with images
- **Verifying SSD Sanitization (https://nvmexpress.org/wp-content/uploads/Session-3-Verifying-SSD-Sanitization_Micron_Toshiba_Final-as-of-4.26.pdf)**