Mpt3sas Linux driver - Binary rpm/source rpm user guide and known limitation

Contents

1. OVERVIEW

This README covers Avago's MPT3SAS Linux driver specific limitation and information. For any OS Distro specific limitation and information please check with OS Vendor support.

2. DRIVER COMPILATION STEPS

Driver source code is placed inside released driver package. Driver source tarball should be with name – mpt3sas-<driver_version>-src.tar.gz

1) Untar driver source tarball-
   #tar -zxvf mpt3sas-<driver_version>-src.tar.gz

2) Go to driver source directory-#cd mpt3sas

3) To compile the driver for distro use the helper script "compile.sh" bundled inside source code-
   #./compile.sh

4) To load and unload the driver for distro use the helper script "load.sh" and "uload.sh" bundled inside source code-
   #./load.sh
   #./uload.sh

3. OS SUPPORT MATRIX

Check os_support.txt file.

4. RELEASE CONTENTS:

For any queries on supported OS matrix, please refer above **SECTION #3** contents. OS Support list in SECTION #3 list out test coverage executed by Avago. MPT3SAS driver is GPLv2 open source driver and source code level support is possible for many linux kernel versions. If you do not find binary level support for your distribution in release contents, please use source rpm method.

For any distribution or supported kernel version, there can be three possible packages –

- Driver update disk (Available under folder disks-xx)
- Precompiled binary (kmod/kmp/rpms).(Available under folder rpms-xx)
- Source rpm. (Available under folder rpms-xx)

5. ERRATA/NOTES AND KNOWN LIMITATIONS

a. Oracle Linux Installation errata:

Installing Driver during Installing for UEK from CD:

For UEK kernels, only the KMODs RPMs are provided, not the DUDs. The reasoning behind this is the OEL installation is using the native Red Hat kernels, not UEK. UEK kernel RPMS can be installed after the basic installation is completed. Please check with Oracle support team w.r.t UEK kernel installation process, limitation and other technical queries which is more of generic and not related to MR Drivers.

b. RPM install dependency issues:

If driver RPM installation fails with kABI checks dependency failure message, installing RPM package the user will need to use the "--nodeps" switch when installing the binary."

Example: rpm -ivh --nodeps kmod-mpt3sas-vxxxxxx_UEK.xxx.rpm

"If "rpm -ivh throw any dependency warning/error"

RPM uses KMOD packaging dependency data to ensure the dependencies are met before installing the binary RPM.
Red Hat maintains a whitelist of kernel symbols which RPM uses to validate against the KMOD binaries. Some symbols may be in the kernel but not on the whitelist which results in a failed binary RPM install. User can use the "--nodeps" switch when installing the binary to skip those whitelist symbol checks or any other dependency."

c. Kernel crash observed on kernels with version >= 4.1

1.Kernel crash observed while creating a second RAID volume.

Issue: Kernel gets crashed while creating 2nd RAID volume. Hence this issue has impact on IR card and not on IT card.

Steps to Reproduce:

>>Boot into OS/kernel with its inbox/out-of-box driver, after discovering HBA and the devices connected to HBA successfully. >>Launch utility (Ex: SAS3IRCU) and create a RAID volume (RAID0/RAID1/RAID10).volume gets created successfully and will be listed from utility.

>>Similarly try to create second RAID volume (RAID0/RAID1/RAID10), Kernel crash is observed while creating second RAID volume.

Expected: One should be able to create MAX 2 RAID volumes with IR card at any point of time successfully , without any kernel crash.

BZ link: Below link has complete details,
https://bugs.launchpad.net/ubuntu/+source/linux/+bug/158132 6

2.Kernel crash observed while unloading the driver keeping enclosure attached.

Issue: Kernel gets crashed while unloading driver keeping enclosure attached. This issue impacts on both IT and IR card.

Steps to Reproduce:

>>Boot into OS/kernel with its inbox/out-of-box driver, after discovering HBA and enclosure with set of drives connected to HBA successfully.
>>Try unloading driver "modprobe -r mpt3sas", kernel crash is observed.

Expected: Driver should unload successfully irrespective of whether enclosure is connected behind IT/IR card.

Workaround Solution: As this issue is due to some "ses" module patch , user may have to unload "ses"(rmmod ses) module first followed by driver unload. Then driver unloads successfully.

BZ link: Below link has complete details,
https://bugzilla.oracle.com/bugzilla/show_bug.cgi BUG:15521

6. INSTALL/REMOVE/UPGRADE DRIVER PACKAGE

There are two packaging formats for binary rpms –

1) RPM(RHEL/SLES/Fedora/OEL uses RPM package)
2) DEB(Ubuntu/Debian uses DEB package) in which driver binary support is provided. More info –
https://en.wikipedia.org/wiki/Deb_%28file_format%29

Please note that after install/remove driver package, system needs to be rebooted to get intended driver loaded or manually remove module and insert (read man page "rmmod" and "modprobe" for more info)

Whenever driver package is installed/uninstalled/upgraded, it is good practice to check output of command #modinfo mpt3sas.

Output should always have updated driver version.

Steps for Driver install/remove/upgrade for .rpm package

1. To install driver RPM, type below command-
```
# rpm -ivh <DRIVER_PACKAGE>.rpm
```

2. To uninstall driver RPM, type below command to check installed driver package name-
```
# rpm -qa | grep mpt3sas
```

Output will give installed mpt3sas RPM packages.
Now type # rpm –e <package to be uninstalled>
3. To upgrade driver RPM, type below command-
```
#rpm -Uvh <DRIVER_PACAKGE>.rpm
```

Steps for Driver install/remove/upgrade for .deb package

1. To install .deb package, execute following command,
`#dpkg -i <DRIVER_PACKAGE>.deb`

2. To verify the status of installed packages then type following command
`#dpkg -s mpt3sas`

3. After installing mpt3sas driver, type below
`#modinfo mpt3sas`

It should show the currently installed version of mpt3sas

4. To use installed DEB driver loaded, restart the machine and type following command to get currently loaded driver version-
`#cat /sys/modules/mpt3sas/version`

This version should be same as driver version of installed driver DEB package.

5. To uninstall mpt3sas package, type below command-
`#dpkg -r mpt3sas`

6. Verify "modinfo mpt3sas" mpt3sas version should be in-box version.

7. STEPS TO GENERATE BINARY RPM FROM SOURCE RPM

There are three variant of source rpms available in this package –

a. Source rpm which use kmodtool interface (RHEL based)
http://rpmfusion.org/Packaging/KernelModules/Kmods2

b. Source rpm which use kmp build interface (SLES based)
https://en.opensuse.org/Kernel_Module_Packages

c. Source rpm which use generic build interface (Create initramfs internally and does not depend upon any external tool)

See **NOTES** section for sample naming convention used for these three flavor of source rpm.

If user doesn't know which source rpm is better for their environment, we recommend trying #c (for any other distro other than Redhat/Novell)
Quick search of "rpm –qa |grep kmod" can provide hint, if kmod tool support is available or not.

To generate binary rpm from source rpm, user should have compilation/build environment to create kernel module, utilities to build RPM(e.g. rpmbuild..).Install **"kernel-devel"** or **"linux-headers"** depending on distro for compilation environment. E.a system where user has yum repo configured, use below.

yum groupinstall "Development Tools"

E.a on Ubuntu user can try installing below missing components.

apt-get install rpm
apt-get install make
apt-get install gcc
apt-get install alien

Exact commands to create build environment would be different across distros so this document is not right place to cover those details. Please refer OS vendor document if needed how to create build/compilation environment for specific OS distro.

Below are steps to generate binary RPM from source RPM-

1. Install source RPM using command <rpm>. Example below-

```
#rpm -ivvh mpt3sas-<driver_version>.src.rpm
```

   Installing above RPM will copy driver SPEC file to specific location (configured as part of rpm package. This path can be different for each OS distribution.)

   To locate SPEC file, check output logs of above source RPM installation (see blue marked gives SPEC file location).

   e.g.

```
[root@localhost tmp]# rpm -ivvvh mpt3sas-06.810.00.02-
98.src.rpm
D: ============== mpt3sas-06.810.00.02-98.src.rpm
..
Updating / installing...
   1:mpt3sas-06.810.00.02-
98################################# [100%]
D: ========== Directories not explicitly included in package:
D:          0 /root/rpmbuild/SOURCES/
D:          1 /root/rpmbuild/SPECS/
D: ==========
D: unknown 100755 1 ( 0, 0) 25
/root/rpmbuild/SOURCES/Module.supported;55a756c8
```

```
D: unknown 100644 1 ( 0, 0)120552
/root/rpmbuild/SOURCES/mpt3sas-06.810.00.02.tar.gz;55a756c8
D: unknown 100644 1 ( 0, 0) 6783
/root/rpmbuild/SPECS/mpt3sas.spec;55a756c8
GZDIO:      17 reads,   127888 total bytes in 0.000585 secs
D: closed   db index       /var/lib/rpm/Name
D: closed   db index       /var/lib/rpm/Packages
D: closed   db environment /var/lib/rpm
```

2. Go to directory where driver SPEC file is copied as part of #1. There must be SPEC file e.g. mpt3sas.spec/lsi-mpt3sas or mpt3sas.spec(driver SPEC file name could be different for different distros, so check SPEC file with megaraid string in its name).

   Check spec file name in above command. It provides the location and spec filename.

3. Build binary RPM from source RPM. Below is command to do same-`#rpmbuild -ba <DRIVER_SPEC_FILE>`

   For Fedora23 onwards:
   `#rpmbuild –ba --define "debug_package %{nil}" <DRIVER_SPEC_FILE>`
   (where “debug_package”(debuginfo) is mandatory on Fedora23 and later versions)

4. Binary RPMs will be available if #3 exits without any error. Go to directory where new binary RPM is generated.

   E.a Snippet of working case –

   --

```
Wrote: /root/rpmbuild/SRPMS/mpt3sas-06.810.00.02-98.src.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/mpt3sas-06.810.00.02-
      98.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/mpt3sas-debuginfo-06.810.00.02-
      98.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.ZbHbmH
+ umask 022
+ cd /root/rpmbuild/BUILD
+ cd mpt3sas-06.810.00.02
+ /usr/bin/rm -rf /root/rpmbuild/BUILDROOT/mpt3sas-06.810.00.02-
      98.x86_64
+ exit 0
```

   --

Step #5 is applicable only for deb package based OS distribution (Ubuntu/Debian).

5. Create driver .deb package from binary RPM, Execute below command to achieve the same-

```
# alien -k --to-deb --scripts <GENRATED_DRIVER_RPM>
```

Now, Install generated binary rpms using steps mentioned at "**Install/Remove/Upgrade driver binary package**"

It is always recommended to verify initramfs image to confirm updated mpt3sas driver is packaged correctly before system reboot. Do not assume that latest driver is loaded after installing rpm generated from source rpm. See OS distribution specific documentation on how to verify initramfs image. E.a FC21 user guide link for reference.

https://docs.fedoraproject.org/en-US/Fedora/21/html/System_Administrators_Guide/sec-Verifying_the_Initial_RAM_Disk_Image.html#bh-Verifying_the_Initial_RAM_Disk_Image_and_Kernel_on_IBM_eServer_System_i

8. HOW TO USE DKMS SOURCE RPM/DEB FOR UBUNTU

Canonical recommends- for Ubuntu, driver support should be provided through DKMS based driver source deb package. We provide single driver source Deb package which should work for all Ubuntu versions supported by us.

We have captured here how to use DKMS enabled driver module. To install DKMS itself, you simply install (or upgrade) from

Ubuntu repo:

```
apt-get install dkms
```

In case you get DKMS Deb/RPM package, then install it using below commands:

For Deb package-

```
dpkg –i dkms-<version>-<release>.noarch.deb
```

For RPM package-

```
rpm -ivh dkms-<version>-<release>.noarch.rpm
```

This is a prerequisite to installing DKMS-enabled module RPMs. To install a DKMS enabled module Deb/RPM, you simply install it like any other Deb/RPM:

For Deb package-

dpkg -i <module>-<version>-<rpmversion>.noarch.deb

For RPM package-
rpm -ivh <module>-<version>-<rpmversion>.noarch.rpm

With a DKMS enabled module Deb/RPM, most of the installation work done by the RPM is actually handed off to DKMS within the RPM/Deb. Generally it does the following:

1. Installs module source into /usr/src/<module>-<moduleversion>/
2. Places a dkms.conf file into /usr/src/<module>-<moduleversion>/
3. Runs: # dkms add -m <module> -v <version>
4. Runs: # dkms build -m <module> -v <version>
5. Runs: # dkms install -m <module> -v <version>

Once the RPM/Deb installation is complete, you can use DKMS to understand which module and which module version is installed on which kernels. This can be accomplished via the command:

dkms status

From here, you can then use the various DKMS commands (eg. add, build, install, uninstall) to load that module onto other kernels.
For more details, please refer URL- https://github.com/dell/dkms

9. MPT3SAS DRIVER SUPPORTED MODULE PARAMETERS

Mpt3sas driver module can accept arguments from the command line. This allows dynamically changing the behavior of the module according to the given parameter, and can avoid the having to indefinitely change/compile the module during a test/debug sessions. Below are the mpt3sas supported module parameters.

Example for using module parameter:

Logging_level is used as an example here.

In case of grub, Add below string at the end of kernel module parameters line in /boot/grub/menu.lst or /boot/grub/grub.conf file and reboot the system.

Eg: mpt3sas.logging_level=0x3f8

In case of grub2, kernel boot parameters are specified in GRUB_CMDLINE_LINUX option in /etc/default/grub file. After adding module parameter in GRUB_CMDLINE_LINUX regenerate grub.cfg and reboot the system.

Step 1: Edit GRUB_CMDLINE_LINUX in /etc/default/grub file and add module parameter to the GRUB_CMDLINE_LINUX as shown below.
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=rhel_dhcp-10-123-20-25/root rd.lvm.lv=rhel_dhcp-10-123-20-25/swap rhgb console=ttyS0,115200 console=tty0 mpt3sas.logging_level=0x3f8"

Step 2: Generate new grub.cfg file and Reboot the system. grub2-mkconfig -o /boot/grub2/grub.cfg **In UEFI-based system:**
grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg

LIST OF MODULE PARAMETERS:

logging_level:
The logging_level feature prints debug messages of the enabled log areas to the kernel message log file /var/log/messages. Below is the bit map for enabling additional logging feature to debug below listed areas.
Default=0

MPT_DEBUG	0x00000001
MPT_DEBUG_MSG_FRAME	0x00000002
MPT_DEBUG_SG	0x00000004
MPT_DEBUG_EVENTS	0x00000008
MPT_DEBUG_EVENT_WORK_TASK	0x00000010
MPT_DEBUG_INIT	0x00000020
MPT_DEBUG_EXIT	0x00000040
MPT_DEBUG_FAIL	0x00000080
MPT_DEBUG_TM	0x00000100
MPT_DEBUG_REPLY	0x00000200
MPT_DEBUG_HANDSHAKE	0x00000400
MPT_DEBUG_CONFIG	0x00000800
MPT_DEBUG_DL	0x00001000
MPT_DEBUG_RESET	0x00002000
MPT_DEBUG_SCSI	0x00004000
MPT_DEBUG_IOCTL	0x00008000
MPT_DEBUG_CSMISAS	0x00010000
MPT_DEBUG_SAS	0x00020000
MPT_DEBUG_TRANSPORT	0x00040000

MPT_DEBUG_TASK_SET_FULL 0x00080000

E.g. mpt3sas.logging_level=0x00000008 => Enables debug logs for EVENTS (MPT_DEBUG_EVENTS 0x00000008).

enable_sdev_max_qd:
This allows the user to set each sdev queue depth to corresponding shost's can_queue value when this module parameter is enabled. This module parameter is disabled by default.

max_sectors:
Driver imposed sector transfer limit. It indicates the maximum number of sectors to transfer per SCSI IO command. Max_sectors range [64 to 32767].
Default=32767 (int).

command_retry_count:

The set number of times the internally SCSI Scan (e.g. Inquiry, TUR, Report Lun etc.) commands will be retired during drive discovery process when drive is hot added. Default=144 (int)

missing_delay:

This field has Device missing delay and Io missing delay as an array of Int.

Device missing delay: The number of seconds the IOC will delay reporting a target device missing after it becomes unavailable. The device will not be reported as missing if it returns before this timer expires.

IO Missing Delay: The number of seconds the IOC will delay replying to SCSI Initiators request messages when the addressed device is missing because of the inability to access the target device.

Note: Device missing delay should be greater than IO Missing Delay. E.g. (mpt3sas.missing_delay=20,20)

host_lock_mode:

When this module parameter is enabled then it will set spinlock named host_lock of Scsi_Host. Default=0 (int)

max_lun:
This sets max_lun field in the Scsi_Host data structure.
Default=16895 (int)

hbas_to_enumerate:

A System might have multiple HBA's. This option enables user to select the HBA's to be managed by the driver.

0 - Enumerates all SAS 2.0, PCIe HBA, SAS3.0 & above generation of HBAs.

1 - Enumerates only PCIe HBA & SAS 2.0 generation HBAs.

2 - Enumerates PCIe HBA, SAS 3.0 & above generation HBAs.

Default=-1 (Enumerates all SAS 2.0, PCIe HBA, SAS 3.0 & above generation HBAs else PCIe HBA, SAS 3.0 & above generation HBAs only) (int)

multipath_on_hba:

Multipath support to add same target device as many times as it is visible to HBA from various paths. Default=0 (Disabled) (int)

disable_eedp:

End-to-end Data Protection is a feature in hard drives that extends error detection to cover the entire path from the computer system to the hard drive media and back.

Default=0 (EEDP support is enabled) (uint)

diag_buffer_enable:

When this module parameter is set then corresponding diag buffers will be registered with the HBA firmware. Here are list of diag buffer that user can register.

Bit fields for diag_buffer_enable

Bit 0 set = TRACE

Bit 1 set = SNAPSHOT

Bit 2 set = EXTENDED

Default=0 (none of the diag buffers will be registered)(int)

disable_discovery:

Setting this module parameter will disables discovery of attached drives.

Default=0 (int)

allow_drive_spindown:

Allows host driver to issue START STOP UNIT (STOP) command to spin down the drives before shutdown or driver unload.

Default=1 (Spindown SSD but not HDD) (uint)

0 => don't Spindown any SATA drives.

1 => Spindown SSD but not HDD

2 => Spindown HDD but not SSD

3 => Spindown all SATA drives

prot_mask:

Permits overriding the host protection capabilities mask. DIF defines the exchange of protection information between initiator and SBC block device (HBA and Disk). DIX defines the exchange of protection information between OS and initiator (OS and HBA).

Bits for enabling DIF/DIX

```
SHOST_DIF_TYPE1_PROTECTION = 1 << 0
SHOST_DIF_TYPE2_PROTECTION = 1 << 1
SHOST_DIF_TYPE3_PROTECTION = 1 << 2
SHOST_DIX_TYPE0_PROTECTION = 1 << 3
SHOST_DIX_TYPE1_PROTECTION = 1 << 4
SHOST_DIX_TYPE2_PROTECTION = 1 << 5
SHOST_DIX_TYPE3_PROTECTION = 1 << 6
```

Default=0x07 (DIX support is disabled and only DIF will be enabled) (int)

protection_guard_mask:

This permits overriding the host protection algorithm mask Available (T10 CRC/ IP Checksum).

All DIX-capable initiators must support the T10-mandated CRC checksum. Controllers can optionally implement the IP checksum scheme which has much lower impact on system performance.

Note that the main rationale for the checksum is to match integrity metadata with data.

```
SHOST_DIX_GUARD_CRC = 1 << 0
SHOST_DIX_GUARD_IP  = 1 << 1
```

Default=3 (int)

issue_scsi_cmd_to_bringup_drive:

Setting this allows host driver to issue SCSI commands internally to bring the drive to READY state. Default=1 (int)

sata_smart_polling:

Setting this feature, allows polling for smart errors on SATA drives.

Default=0 (uint)

max_queue_depth:

User can use this to set hba queue depth to be used by the controller.
Max controller queue depth can be up to 30000 (int)

max_sgl_entries:
This allows user to specify the maximum number of Scatter-Gather entries per I/O.
Default=128 in most kernels. (Range 16 to 256) (int)

msix_disable:
Setting 1 disables MSIX routed interrupts and uses legacy interrupts.
Default=0 (int)

smp_affinity_enable:
This defines the CPU cores that are allowed to execute the ISR for that IRQ.
Setting this enables SMP affinity feature. Enable/Disable (1/0)
Default=1 Enabled (int)

max_msix_vectors:
Controls the number of msix vectors to be used by mpt3sas driver.
SAS3.0 controllers like Invader supports max 96 msix vectors
SAS3.5 products like Ventura supports max 128 msix. (int)

irqpoll_weight:
irq poll weight is used as the budget in processing
of reply descriptors from reply descriptor post queues. Also once irqpoll_weight numbers of reply descriptors are continuously processed in ISR context then driver will quit from the ISR and invoke the irq polling thread to process the remaining reply descriptors.
Default=one fourth of HBA queue depth. (int)

mpt3sas_fwfault_debug:
Setting this enables detection of firmware fault and halts firmware. Usually this is set to collect the HBA FW logs during firmware fault.
Default=0 (int)

perf_mode:
This module parameter allows the user to select one of the following performance mode (only for Aero/Sea Generation) options,
      0 - balanced: high iops mode is enabled & interrupt coalescing is enabled only on high iops queues,

1 - iops: high iops mode is disabled & interrupt coalescing is enabled on all queues,
2 - latency: high iops mode is disabled & interrupt coalescing is enabled on all queues with timeout value 0xA, default - on Intel architecture, default perf_mode is 'balanced' and in others architectures the default mode is 'latency'. (int)

10. NOTES

1. In case of SuSE and similar OS distribution which use kmp tool for initramfs, we recommend to edit file - /etc/sysconfig/kernel. Append <mpt3sas> before installing binary rpm.

```
INITRD_MODULES="ata_piix ata_generic mpt3sas"
```

2. If you see initramfs failure while rpm installation, try generating inird manually. Just make sure that <modinfo mpt3sas> provide the expected driver version.
3. Do not try to install/upgrade from source rpm generated rpm binaries along with pre-compiled binary provided by Avago. Naming convention as part of precompiled binary and source rpm based rpm differs, and that is reason user should stick with one method to avoid compatibility issue.

   Source rpm is quickest method to deploy driver on many supported kernel versions, whereas precompiled binary is specific to kernel version or set of kernel versions.

   Example –

   Source rpm based binary will be generated in below format -

   kmp_rpm:
   lsi-mpt3sas-kmp-default-11.255.01.00_3.0.76_0.11-99.x86_64.rpm

   kmod_rpm:
   kmod-mpt3sas-11.255.01.00-61.x86_64.rpm

   generic_rpm:
   mpt3sas-11.255.01.00-98.x86_64.rpm

4. On System with large number of CPU core and LSI’s SAS3 controllers, on repeated load and unload of mpt3sas driver module, if kernel fails to allocate the memory requested for higher queue depth, we can observe that the loading of mpt3sas module fails. Below messages will be logged to /var/log/messages,

   mpt3sas0: chain_lookup: __get_free_pages failed

mpt3sas0: Reduce the module parameter max_queue_depth to a value lower than ("CURRENT_VALUE_OF_QUEUE_DEPTH") and retry.

The work-around for this issue is to load mpt3sas driver with module parameter max_queue_depth set to value less than CURRENT_VALUE_OF_QUEUE_DEPTH.

The max_queue_depth module parameter could be set as follows

a. While loading the driver

modprobe mpt3sas max_queue_depth=NEW_VALUE_OF_QUEUE_DEPTH (if driver rpm is already installed)

(Or)

insmod mpt3sas.ko max_queue_depth=NEW_VALUE_OF_QUEUE_DEPTH (if you have a mpt3sas.ko file)

b. If driver is in ramdisk, then in RHEL5/SLES/OEL5 OS, following line has to be added in /etc/modprobe.conf and reboot the system
options mpt3sas max_queue_depth=NEW_VALUE_OF_QUEUE_DEPTH

(Or)

Add below word at the end of kernel module parameters line in /boot/grub/menu.lst or /boot/grub/grub.conf file and reboot the system

mpt3sas.max_queue_depth=NEW_VALUE_OF_QUEUE_DEPTH

5. When Target Reset is issued using below command to DIF type2 drive present in the topology then kernel panic is observed on few kernels.

   echo 4 > sys/class/scsi_host/host(number)/task_management

   The users can apply below patch if applicable otherwise can check with the kernel vendors for the appropriate patch

   http://marc.info/?l=linux-scsi&m=135186352200668&q=raw

6. By default mpt3sas driver will disable DIX support(prot_mask =0x07), user can enable this feature by setting "prot_mask = 0x7f" module parameter while driver load.

   The "prot_mask" module parameter could be set as follows:

a. While loading the driver
modprobe mpt3sas prot_mask=0x7f (if driver rpm is already installed)
(Or)
insmod mpt3sas.ko prot_mask=0x7f (if you have a mpt3sas.ko file)

b. If driver is in ramdisk, then in RHEL/SLES/OEL OS, following line has to be added in /etc/modprobe.conf and reboot the system options
mpt3sas prot_mask=0x7f
(Or)
Add below word at the end of kernel module parameters line in /boot/grub/menu.lst or /boot/grub/grub.conf file and reboot the system

mpt3sas.prot_mask=0x7f

7. On some kernels when mq is enabled then the user may observe kernel panic such as 'NULL pointer dereference', 'protection fault' etc. when user perform expander reset or driver unload operations. This is a kernel issue as 'scsi_host_find_tag' API is providing some stale requests pointers when driver loops from smid one to hba queue depth after some drives are removed and added back. Below patch will fix this issue, https://www.spinics.net/lists/linux-scsi/msg126041.html

---

Special Notes for This Build:

---

8. i) This release order will include binaries which compiled with retpolined GCC options,

   * On OS GA release for which retpoline was not supported and also on retpoline patched kernels which are released post OS GA(RH 7.4/7.3/7.2, SLES12.x/11.x, OEL6.x/7.x) driver is compiled with GCC options "-mindirect-branch=thunk-inline -mindirect-branch-register"

   * On OS GA release for which retpoline was supported(RHEL7.5), SLES15 BRCM doesn't add any extra retpoline GCC flags during driver compilation, since OS itself will add "-mindirect-branch=thunk-extern" GCC option by default.

ii) Some information on driver binaries built for OS GA kernels(which were released before retpoline support):

   * Both driver RPMs and DUDs are compiled with GCC option: "-mindirect-branch=thunk-inline -mindirect-branch-register" They are safe from Spectre v2 vulnerabilities.
   * "modinfo mpt3sas.ko" will show "retpoline :Y" for driver module built with retpoline support.

* Retpoline kernels have "CONFIG_RETPOLINE=y" in kernel config file.
* Driver binaries (DUD and RPM) will work for both OS GA as well as retpoline kernels.

iii) SLES 15 OS installation with mpt3sas Driver:

* While installing the SLES15 OS, If Broadcom's IT HBA is not attached to the system then mpt3sas.ko binary won't be part of the initrd image.
* If user installs the driver rpm on this environment then installed mpt3sas.ko won't be part of initrd image.
* So it recommended attaching the IT HBA card while installing the SLES15 OS.

iv) Installing rpm in OEL 6.10.

There is an OS (Oracle Linux 6.10) issue found in one of the installer scripts "weak-modules".

During driver installation updated initramfs with latest driver is not getting generated. Due to this after reboot still old driver is getting loaded. As a workaround, after installation
regenerate new initramfs manually using below commands:

a) First make a backup of the existing initramfs

```
# cp /boot/initramfs-4.1.12-124.16.4.el6uek.x86_64.img
/boot/initramfs-4.1.12-124.16.4.el6uek.x86_64.img.back
```

b) Create new initramfs:

```
# dracut -f
```

V) Most of the Linux distro’s config file has the required modules enabled for compiling mpt3sas driver. If Linux Kernel is compiled with customized config, make sure below modules are enabled in the config file.

```
CONFIG_CHR_DEV_SG      for sg.ko
CONFIG_RAID_ATTRS      for raid_class.ko
CONFIG_SCSI_SAS_ATTRS for scsi_transport_sas.ko
CONFIG_BLK_DEV_SD      for sd_mod.ko
CONFIG_IRQ_POLL        for IRQ polling library
```

11. SIGNING SOURCE RPM

All Source RPMs are signed/encrypted with GnuPG public-private encryption scheme to ensure the authenticity of the source RPMs. GPG public-private key pair is a 4k bit long with RSA encryption algorithm and all SRPMS are signed with the same. The private key is only held by Broadcom and associated public key is kept on the Broadcom website. Below is the verification guide link which contains the detailed steps to verify the source RPM and public key link to download the public key file.

Verification guide link:
VERIFICATION-GUIDE

Public key link:
PUBLIC-KEY

Note: Verification guide also contains the public key link.

12. SIGNED BINARIES RHEL/SLES

Broadcom/DCSG provides unsigned binaries for all supported OS's. In addition to unsigned binaries, Broadcom/DCSG also provides signed binaries for SLES and RHEL. The release order has a folder prefixed with signed for signed driver binaries.
Ex.
signed_sles12 => signed sles12 binaries;
signed_sles15 => signed sles15 binaries;
sles12 => unsigned sles12 binaries;
sles15 => unsigned sles15 binaries;

Broadcom/DCSG will sign SLES and RHEL driver binaries with DCSG owned public-private key pair. The private key is only held by Broadcom and associated public key will be provided as part of Linux driver RO (release order package) with name “DCSG00411462_selfcert.der”.

To use signed driver, Please refer OS vendor documentations.

SUSE:
https://drivers.suse.com/doc/Usage/Secure_Boot_Certificate.html#secure-boot-certificate