The nodev mount option and related security considerations.

This document **(000020660)** is provided subject to the disclaimer at the end of this document.

Situation

Some security scanning software is known to flag a filesystem mounted without the nodev option as a security issue.

This leads to customers asking why nodev is not automatically present and active on all mounts, both system created and user created. Specifically, they have concerns that a filesystem mounted without the nodev option may mean that the mount in question is less secure than if it had the nodev option added.

Resolution

For many file systems, possibly the majority, using the nodev option probably does make a mount more secure. However, not using it allows access to certain functionality that could be helpful in a small number of use cases.

Cause

Security scanning software may flag a mounted filesystem that was mounted without the 'nodev' option as a security risk.

Additional Information

According to the man page for mount, the nodev option applies the following:

> “Do not interpret character or block special devices on the filesystem”.

This poses the question:

"What is a character or block special device and how could one be used to make a mount less secure or make a mount more useful ?".

A character or block special device permits access to a file that has a backing device that isn't part of the file-system it is contained on. A simple example is the “disk” objects on the /dev file-system. Here are long directory listings for an NVME device file and a USB hard disk device file:

```
# ls -la /dev/nvme0n1
brw-rw---- 1 root disk 259, 0 May 20 19:58 /dev/nvme0n1
```

```
# ls -la /dev/sdb
brw-rw---- 1 root disk 8, 16 May 20 19:58 /dev/sdb
```

Each is a name on the dev file-system and, if you have permission, you can read or write to it via the file on dev, which would let you write what you like to a storage device. On SUSE Linux versions, the default permissions are read/write access for user root and group disk but no permissions for others. The disk group usually has very limited user members. The result is that it usually requires being the root user to perform mounts. Any partition table entries also have entries on the /dev file-system, usually with names p1, p2, p3, etc. per-partition:

Example of block device files:

```
# ls -la /dev/nvme0n1
brw-rw---- 1 root disk 259, 0 May 20 19:58 /dev/nvme0n1
brw-rw---- 1 root disk 259, 1 May 20 19:58 /dev/nvme0n1p1
brw-rw---- 1 root disk 259, 2 May 20 19:58 /dev/nvme0n1p2
brw-rw---- 1 root disk 259, 3 May 20 19:58 /dev/nvme0n1p3
```

There are a couple of unusual things about these directory entries:

For most directory listings you will either see '–' or 'd' for the first character of the flags at the start of the line. The 'd' flag would be used for a directory and '-' would be used for a normal data file. Disk devices on the /dev file-system have the 'b' flag, which indicates they are block storage device special files (a reference to a block storage device).

The device files have a strange looking file-size (in the case of /dev/nvme0n1 it is '259, 0'). The numbers don’t indicate a size, they indicate the kernel device Major ID (259) and Minor ID (0) numbers.

Example of tty device file:

```
# l /dev/tty0
crw--w---- 1 root tty 4, 0 May 20 19:58 /dev/tty0
```

The 'c' flag that is used to denote a character device special file, e.g.:

Again, the file size looks unusual (for /dev/tty0 it is '4,0') and again the numbers don’t indicate a size, they indicate the kernel device Major ID (4) and Minor ID (0) numbers.

Normal use of the block storage devices is not via direct access to these kernel devices, it is via file-systems, a device, or a partition that has been formatted and is then mounted into the system directory structure at some location. This is relatively secure but it is open to possible abuse or usage, if it is on a file-system that is mounted without the nodev option. You can get a list of the active file-systems that use nodev by default by looking at the contents of /proc/filesystems:

```
# cat /proc/filesystems
nodev    sysfs
nodev    tmpfs
nodev    bdev
nodev    proc
nodev    cgroup
nodev    cgroup2
nodev    cpuset
nodev    devtmpfs
nodev    debugfs
nodev    tracefs
nodev    securityfs
nodev    sockfs
nodev    bpf
nodev    pipefs
nodev    ramfs
nodev    hugetlbfs
nodev    devpts
nodev    autofs
nodev    mqueue
nodev    pstore
nodev    efivarfs
         btrfs
nodev    configfs
         fuseblk
nodev    fuse
nodev    fusectl
         vfat
         xfs
```

What stands out is that the types of file-systems that don't use nodev by default are those intended for normal user data files, e.g. btrfs, xfs, etc. The special file-systems default to using nodev.

What does mounting without nodev let you use or abuse? An example of this follows. One reduces system security (abuse) and the other doesn't reduce security (use).

CASE OF ABUSE

Consider a system where there is no root or sudo user access, where access security is well managed, but there is physical access to the system. The following steps could be used to abuse a system that has a filesystem mounted without nodev, resulting in damage the system:

1. In a non-root ssh session to the secure system, use the mount command to get the devnode names for the mounted file-systems:

```
# mount | grep "\s/\s"
/dev/nvme0n1p2 on / type btrfs
(rw,relatime,ssd,space_cache,subvolid=266,subvol=/@/.snapshots/1/snapshot)
```

2. Use ls -l to look at the devnodes for the disk containing the root file-system:

```
# ls -la /dev/nvme0n1*
brw-rw---- 1 root disk 259, 0 May  5 09:05 /dev/nvme0n1
brw-rw---- 1 root disk 259, 1 May  5 09:05 /dev/nvme0n1p1
brw-rw---- 1 root disk 259, 2 May  5 09:05 /dev/nvme0n1p2
brw-rw---- 1 root disk 259, 3 May  5 09:05 /dev/nvme0n1p3
```

Note the Major, Minor device ID for the partition containing the root file-system, 259, 2.

3. Look at the contents of /proc/filesystems and find one that does not use nodev by default:

```
# cat /proc/filesystems | grep -v nodev
btrfs
fuseblk
vfat
xfs
```

4. Choosing xfs for no special reason, exit the ssh session to the secure system and continue the process by working on a different 'insecure' system where the root password is known.

5. Plug-in a USB thumb-drive and set it up with an XFS file-system (just a data file-system, no need to be bootable) and mount it on this system.

6. Remembering the block device ID for the partition containing the root file-system on the secure system, open a shell, su to root, change directory to the location where I mounted the XFS file-system from the thumb drive and use the following:

```
# mknod -m 666 wretched b 259 2
```

7. Continue by using the following to examine the block device:

```
# ls -la wretched
brw-rw-rw- 1 root root 259, 2 May 10 11:55 wretched
```

It seems to be a block device special file named 'wretched' and it refers to the block device with ID 259, 2 (which was the ID of the block device containing the root file-system on the secure system)

Although it has root as the user and group, importantly, it also has read and write permission for 'others'. That came from using the 666 as the parameter to -m when using mknod.

8. umount the file-system from the thumb-drive, eject the drive and unplug it from my the insecure system

9. As previously stated, physical access to the secure system is available and generally,
Linux automounts file-systems from hot-plug devices like USB disks, including thumb drives. So, access the physical location of the secure system and plug-in the USB thumb drive, then return to the insecure system.

10. Start a new ssh session to the secure system. Even though the ssh session is not authenticated as root, no root access is present, and no membership of disk 'group', use the following command:

```
# mount
…
/dev/sdc1 on /run/media/Thumber type xfs
(rw,nosuid,noexec,relatime,attr2,inode64,logbufs=8,logbsize=32k,noquota,user)
```

Note that nodev is not present in the mount flags

11. Change the directory to the mount-point of the thumb drive and list the contents:

```
# ls -la
brw-rw-rw- 1 root root 259, 2 May 10 11:55 wretched
```

12. At this point, it is possible to completely erase the root file-system on the secure system, by writing data from urandom to the physical block device containing the root file-system:

```
# dd if=/dev/urandom of=./wretched
```

NOTES:

* Using urandom is no more effective than using /dev/zero, it just looks like corruption disaster when reviewed after failure, whereas all zeros does look like intentional damage.

* If the xfs file-system on the thumb drive was mounted with nodev, the ability to destroy the root filesystem would be prevented (Do not interpret character or block special devices on the filesystem). In other words, the wretched node device file would not be interpreted and could not be used to over-write.

* NFS is intentionally a special case that mounts with nodev by default. The reason is that the place where the mount is used is a NFS client and the actual file-system is on a NFS server. The permissions and userbase on the server isn't required to match those on the client. So, an administrator on the NFS server could use mknod in the manner described above to create block device special files available to users on NFS clients where the NFS file-system is mounted.

* Whether or not nodev is used by default is a feature of the code for the file-system so you can't modify the presence of nodev as a global feature or for a given entry in /proc/filesystems on-demand. It would require modifying kernel code and re-building the kernel.

* The proper approach to securing against the above abuse is to consider knowledge of it as part of system administration and prevent hot-plugging of disks or the automounting of them, to only use /etc/fstab mounts, to include nodev in the flags for all of them and when absolutely having to use mount at the command-line, consider the use of nodev as a requirement in all cases.

* Using a character special file (mknod with the 'c' rather than 'b' option) it might be possible to use the above method to create access to hardware device control access ports: PCI controllers, USB controllers, storage controllers, graphics controllers, network controllers. In that case it might be possible to do damage with the above method such that a system locks up or
becomes unresponsive.

* For cases like the above, security software that objects to file-systems mounted without nodev has a good case. However, there are good reasons to mount file-systems without nodev…

CASE OF USE

Normally, there will be very few (if any) device special files on the /dev file-system. There are useful applications of essentially the same process described as abuse above, although they need to include some minor changes to improve security.

1. Consider a system intended to record serial data, i.e. data that continues to be generated over time but once obtained will never be replaced. A simple case would be scientific data that is regularly generated and all data over all time is useful and no previously obtained value is ever replaced. For example, brightness of the sky every minute all day in a location where there is concern about smoke or other dirt in the air. If a record of all data existed for such a case then it would be possible to study it forward, backward and paused. It could be viewed forward or backwards in fast or slow-motion, it could be single stepped forward or backwards. There are similar use cases and not all are scientific.

2. Data like this can be very small individual records but over time can become very large entries. Placing this on a file-system incurs a significant overhead in multiple ways and fragmentation can be extremely limiting to useful capacity. If the file-system is only used for this data in a single file, it's

a large overhead which may be hard to justify in terms of data value, processing time, running costs etc.

3. Instead, the data could be written to a disk (or to a single partition) at a much higher density than would be obtained as a file on a file-system but that would require that the data gathering application would have to run as root or be a member of the disk group.

4. However, if a local data file-system is mounted without the use of nodev then it is possible to use mknod to create a block device special file referring to the recorded data disk, but limit the rights, giving the application the ability to run as a non-root user and record data with no overhead to it’s own storage device, e.g.:

```
# mknod -b 640 b 258 0 /usr/share/smogrecord
# chown recorder users /usr/share/smogrecord
# ls -la /usr/share/smogrecord
brw-r----- 1 recorder users 258, 0 May 20 09:10 smogrecord
```

In this case we have what looks like a block device special file contained in /usr/share (probably on the root file-system) referring to block device 258, 0. That probably indicates a whole block device, with zero being the Minor ID number. However, it’s only writable by a user named recorder. It can be read by members of the group users and no others have access.

5. This would permit operating the system where the data recording application is run as user “recorder” and is the only use for that user. The data is then recorded by writing to a dedicated block device when data is available. Meanwhile, all members of group users (probably all other users on the system) are able to browse the data directly without damaging it and perform their analyses in parallel with each other and with the recording of new data.

* With proper administration (as described in the NOTES for the abuse example), the use of a file-system without nodev in this way does not reduce security, but the utility of a non-root user writing to a storage device is very useful and permits extremely high data density and performance versus attempting the same operations in a single file located on a file-system. File-system caching and compression may create reasons to debate that assertion but the simplicity of the device base approach is very helpful for serial data recording applications.

Disclaimer

Document ID:
000020660

Creation Date:
23-May-2022

Modified Date:
25-Sep-2025

SUSE Linux Enterprise Server

< Back to Support Search (https://www.suse.com/support/kb/)

SUSE Support Forums

Get your questions answered by experienced Sys Ops or interact with other SUSE community experts.

Join Our Community (https://forums.suse.com/)

Support Resources

Learn how to get the most from the technical support you receive with your SUSE Subscription, Premium Support, Academic Program, or Partner Program.

- **SUSE Customer Support Quick Reference Guide(https://links.imagerelay.com/cdn/3404/ql/731401d3cf**
- **SUSE Technical Support Handbook(/support/handbook/)**
- **Update Advisories(/support/update/)**

Support FAQ (/support/faq/)

Open an Incident

Open an incident with SUSE Technical Support, manage your subscriptions, download patches, or manage user access.

- **Report a Software Vulnerability(/support/security/contact/)**

Go to Customer Center (https://scc.suse.com/login)