Grsecurity/Appendix/Capability Names and Descriptions

This table lists all standard Linux capabilities and one special capability related to grsecurity. With capabilities, the system is divided into logical groups that may be individually granted to, or removed from, different processes. See Capability Restrictions for more information.

Capability Name	Meaning
CAP_ALL	CAP_ALL is not a real capability, but was coded into `gradm` to represent all capabilities. Therefore to denote dropping of all capabilities, but CAP_SETUID, -CAP_ALL and +CAP_SETUID would be used.
CAP_CHOWN	In a system with the [_POSIX_CHOWN_RESTRICTED] option defined, this overrides the restriction of changing file ownership and group ownership.
CAP_DAC_OVERRIDE	Override all DAC access, including ACL execute access if [_POSIX_ACL] is defined. Excluding DAC access covered by CAP_LINUX_IMMUTABLE.
CAP_DAC_READ_SEARCH	Overrides all DAC restrictions, regarding read and search on files and directories, including ACL restrictions, if [_POSIX_ACL] is defined. Excluding DAC access covered by CAP_LINUX_IMMUTABLE.
CAP_FOWNER	Overrides all restrictions about allowed operations on files, where file owner ID must be equal to the user ID, except where CAP_FSETID is applicable. It doesn't override MAC and DAC restrictions.
CAP_FSETID	Overrides the following restrictions, that the effective user ID shall match the file owner ID, when setting the S_ISUID and S_ISGID bits on that file; that the effective group ID (or one of the supplementary group IDs) shall match the file owner ID when setting the S_ISGID bit on that file; that the S_ISUID and S_ISGID bits are cleared on successful return from `chown(2)` (not implemented).
CAP_KILL	Overrides the restriction, that the real or effective user ID of a process, sending a signal, must match the real or effective user ID of the process receiving the signal.
CAP_SETGID	▪ Allows `setgid(2)` manipulation. ▪ Allows `setgroups(2)`. ▪ Allows forged gids on socket credentials passing.
CAP_SETUID	▪ Allows `set*uid(2)` manipulation (including fsuid). ▪ Allows forged pids on socket credentials passing.

CAP_SETPCAP	Without VFS support for capabilities: ▪ Transfer any capability in your permitted set to any pid, remove any capability in your permitted set from any pid. With VFS support for capabilities (neither of above, but) ▪ Add any capability from current's capability bounding set to the current process' inheritable set ▪ Allow taking bits out of capability bounding set. ▪ Allow modification of the securebits for a process.
CAP_LINUX_IMMUTABLE	Allow modification of S_IMMUTABLE and S_APPEND file attributes.
CAP_NET_BIND_SERVICE	▪ Allows binding to TCP/UDP sockets below 1024. ▪ Allows binding to ATM VCIs below 32.
CAP_NET_BROADCAST	Allow broadcasting, listen to multicast.
CAP_NET_ADMIN	▪ Allow interface configuration. ▪ Allow administration of IP firewall, masquerading and accounting. ▪ Allow setting debug option on sockets. ▪ Allow modification of routing tables. ▪ Allow setting arbitrary process / process group ownership on sockets. ▪ Allow binding to any address for transparent proxying. ▪ Allow setting TOS (type of service). ▪ Allow setting promiscuous mode. ▪ Allow clearing driver statistics. ▪ Allow multicasting. ▪ Allow read/write of device–specific registers. ▪ Allow activation of ATM control sockets.
CAP_NET_RAW	▪ Allow use of RAW sockets. ▪ Allow use of PACKET sockets.
CAP_IPC_LOCK	▪ Allow locking of shared memory segments. ▪ Allow `mlock` and `mlockall` (which doesn't really have anything to do with IPC).
CAP_IPC_OWNER	Override IPC ownership checks.
CAP_SYS_MODULE	Insert and remove kernel modules – modify kernel without limit.
CAP_SYS_RAWIO	▪ Allow ioperm/iopl access ▪ Allow sending USB messages to any device via */proc/bus/usb'*
CAP_SYS_CHROOT	Allow use of `chroot()`.
CAP_SYS_PTRACE	Allow `ptrace()` of any process.
CAP_SYS_PACCT	Allow configuration of process accounting.
CAP_SYS_ADMIN	

	- Allow configuration of the secure attention key. - Allow administration of the random device. - Allow examination and configuration of disk quotas. - Allow configuring the kernel's syslog (printk behaviour). - Allow setting the domainname. - Allow setting the hostname. - Allow calling `bdflush()`. - Allow `mount()` and umount(), setting up new smb connection. - Allow some autofs root ioctls. - Allow nfsservctl. - Allow VM86_REQUEST_IRQ. - Allow to read/write pci config on alpha. - Allow irix_prctl on mips (setstacksize). - Allow flushing all cache on m68k (sys_cacheflush). - Allow removing semaphores. Used instead of CAP_CHOWN to "chown" IPC message queues, semaphores and shared memory. - Allow locking/unlocking of shared memory segment. - Allow turning swap on/off. - Allow forged pids on socket credentials passing. - Allow setting readahead and flushing buffers on block devices. - Allow setting geometry in floppy driver. - Allow turning DMA on/off in xd driver. - Allow administration of md devices (mostly the above, but some extra ioctls). - Allow tuning the ide driver. - Allow access to the nvram device. - Allow administration of apm_bios, serial and bttv (TV) device. - Allow manufacturer commands in isdn CAPI support driver. - Allow reading non–standardized portions of pci configuration space. - Allow DDI debug ioctl on sbpcd driver. - Allow setting up serial ports. - Allow sending raw qic–117 commands. - Allow enabling/disabling tagged queuing on SCSI controllers and sending arbitrary SCSI commands. - Allow setting encryption key on loopback filesystem. - Allow setting zone reclaim policy.
CAP_SYS_BOOT	- Allow use of `reboot()` - Allow use of `kexec()` syscall
CAP_SYS_NICE	- Allow raising priority and setting priority on other (different UID) processes.

	▪ Allow use of FIFO and round–robin (realtime) scheduling on own processes and setting the scheduling algorithm used by another process. ▪ Allow setting cpu affinity on other processes.
CAP_SYS_RESOURCE	▪ Override resource limits. Set resource limits. ▪ Override quota limits ▪ Override reserved space on ext2 filesystem ▪ Modify data journaling mode on ext3 filesystem (uses journaling resources). NOTE: ext2 honors fsuid when checking for resource overrides, so you can override using fsuid too. ▪ Override size restrictions on IPC message queues. ▪ Allow more than 64Hz interrupts from the real–time clock. ▪ Override max number of consoles on console allocation. ▪ Override max number of keymaps.
CAP_SYS_TIME	▪ Allow manipulation of system clock. ▪ Allow `irix_stime` on mips. ▪ Allow setting the real–time clock.
CAP_SYS_TTY_CONFIG	▪ Allow configuration of tty devices. ▪ Allow `vhangup()` of tty.
CAP_MKNOD	Allow the privileged aspects of `mknod()`.
CAP_LEASE	Allow taking of leases on files.
CAP_AUDIT_WRITE	Allow emitting auditing messages.
CAP_AUDIT_CONTROL	Allow administration of the kernel's auditing system.
CAP_SETFCAP	Allow the setting of file capabilities.
CAP_MAC_OVERRIDE	Override MAC access. The base kernel enforces no MAC policy. An LSM may enforce a MAC policy and if it does and it chooses to implement capability based overrides of that policy, this is the capability it should use to do so.
CAP_MAC_ADMIN	Allow MAC configuration or state changes. The base kernel requires no MAC configuration. An LSM may enforce a MAC policy, and if it does and it chooses to implement capability based checks on modifications to that policy or the data required to maintain it, this is the capability it should use to do so.
CAP_SYSLOG	Allow configuring the kernel's syslog (printk behaviour).
CAP_WAKE_ALARM	Allow triggering something that will wake the system.

Retrieved from "https://en.wikibooks.org/w/index.php?title=Grsecurity/Appendix/Capability_Names_and_Descriptions&oldid=3305214"

This page was last edited on 28 September 2017, at 16:02.