Engineering Secure Software

LINUX PERMISSIONS

Linux File Permissions

- Each file and directory has bits for..
  - Read, Write, Execute: rwx
  - Files: works as it sounds
- Directories:
  - r → “can list files in a directory” (but not read a given *file*)
  - x → “read a file if you know the name” (easy if directory also has read)
  - w → “can create, change, delete files in directory”
- Thus, you may only read a file IFF you:
  - Have read permissions to the file AND
  - Have execute permissions to that file’s directory
- Files & Directories have 3 levels:
  - Owner, Group, and Everyone Else
  - aka. User, Group, Other: ugo

- List permissions of a file: `ls -l`
  with some info removed from below in [...]

```
drwx--x--- […] andy faculty […] .
drwx------ […] andy faculty […] ..
-rwxrwxrwx […] andy faculty […] allopen
drwx------ […] andy faculty […] mydir
-rw------- […] andy faculty […] myfile
-rwx------ […] andy faculty […] myprog
drwxrwx--- […] andy faculty […] ourdir
-rwxrwx--- […] kenn faculty […] ourprog
```

- Can `kenn` execute `myprog`?
  No, because of file permissions.
- Can both `andy` and `kenn` execute `ourprog`?
  Yes. Everyone in `faculty` can read and execute `myprog`
- Can `kenn` read `ourprog`?
  Yes. Directory has group x, file has group r
- Can `kenn` list to find `ourprog`?
  No. Directory listing is off.

chmod

- Command to change permissions
  - *Set (=)*, *Add (+)*, *Remove (-)*
  - Set user to rw, not x: `chmod u=rw .`
  - Set user, groups to only rw: `chmod ug=rw .`
  - Set ug to only rx, recursively: `chmod -R ug=rx .`
  - Add “groups can rw”: `chmod g+rw .`
  - Add “others can rw”: `chmod o+rw .`
  - Add “everyone can read”: `chmod a+r .`
  - Remove write from group: `chmod g-w .`
- Octal notation
  - e.g. `chmod 755 file.txt`
  - Good for setting, not adding/removing
  - 1,2,3 are never used

rwx	---	--x	-w-	-wx	r--	r-x	rw-	rwx
Binary	000	001	010	011	100	101	110	111
Decimal	0	1	2	3	4	5	6	7

umask

- When a file is created…
  - **U**ser **mask** (`umask`) is consulted for permissions
  - Owner = user who created the file
  - Subtract octally
    - from 666 for files
    - from 777 for directories
    - e.g. `666-022=644, or rw-r--r--`
  - nitron.se.rit.edu default: `rw-------` (or `077`)
  - Common default: `rw-r--r–` (or `022`)
  - Common umask shared group stuff: `rw-rw-r–` (or `007`)
- Programs can change their own umask
  - Blessing for good developers
  - Curse for system administrators

`setuid`, `setgid`

- When executing, ordinarily…
  - OS ignores the owner of the program
  - Runs the program as you
  - (assuming you have permissions, of course)
  - e.g. `prog.sh` owned by root gets run as you
- `setuid` and `setgid` bits on files
  - `chmod ug+s ./prog.sh`
  - "If you can execute this, it's executed as the owner's rights, not as the executing user's rights"
  - Files owned by root should *never* have this set
  - Different from the "sticky bit" (not covered here)
- `setuid` and `setgid` bits on directories
  - `setuid` on directories is ignored in Linux
  - `setgid` means new files inherit the group ID (like umask)

For Example

```
larry$ umask
077
larry$ mkdir dir
larry$ ls –l dir/
drwx------ … larry stooges … dir
larry$ touch dir/file.sh
larry$ ls –la dir/
-rwx------ … larry stooges … .        [./dir]
-rwx------ … larry stooges … ..       [./dir/..]
-rw------- … larry stooges … file.sh
larry$ ./dir/file.sh
bash: ./dir/file.sh: Permission denied
larry$ chmod –R ug+x .

curly$ ./dir/file.sh
[Success!]
curly$ ls –l ./dir
bash: ./dir/: Permission denied
```

Beware of sudo

- He sees you when you're sleeping, he knows when you're awake, he's copied on /var/spool/mail/root, so be good for goodness' sake.
- Source: http://xkcd.com/838