ITEC 345: Secure operations on the operating system (UNIX/Linux)

*Most of the tasks here need to be performed with root privileges, using sudo.*

*E.g., sudo useradd executes the command useradd with root privileges.*

*Whenever you are denied permission, ask yourselves whether this is because you don’t have root privileges. Use sudo to invoke your root privileges.*

*Of course it’s good practice to* only *invoke sudo when needed, to protect yourself from accidentally affecting other users when you didn’t mean to.*

Warning: Please note that you are still bound by the White Hat agreement and the rules of Radford University, the State of Virginia, and the U.S government.

**The following guide lists some of the basic operations that security administrators must perform when securing UNIX/Linux systems.**

In addition to just using the commands mentioned here, you will also want to frequently consult their man pages, so that you understand what they do (and understand the output of the commands).

**Some basics (read this first): To perform system admin commands, you need to perform some tasks as root. There are three ways to be a root:**

1. **Login as root:** this is not a recommended approach. When you login as a root, all the program you execute (e.g., browser) also run with root privileges. So any malicious code that you download will now run with root privileges.
2. **Login as a regular user and use “su” to become root:** You can become a root, by using the program **su (for superuser)** inside a terminal. This limits the root privileges to only those program that are started from that terminal. All other programs run with the user’s privileges.
3. **Use sudo:** sudo allows you to run individual programs with root privileges. This is the best way to get root privileges as it follows the *principle of least privilege:* Only the program that you run with sudo have root privileges and the privileges are removed once the program exits. You can run any program with root privileges using sudo. E.g., sudo /bin/ls (will execute ls with root privileges).

So as a first security administrative task most administrators lock the “root” account. This prevents (1) and (2) above from happening (i.e., no one can login as root). Users are then forced to use sudo (option 3). If you want a terminal with root privileges you can execute “sudo su”. Most of the following tasks need to be executed with sudo.

1. **Create new user accounts and their home directories.** 
   1. **Program to use:** useradd.
      1. Hint: Run “man useradd” and identify the options to use to create the home directory when creating the users.
   2. Verify that all the users were created along with their home directories.
      1. Open the file /etc/passwd**.** You can use the editor “pico” to open the file. Alternatively, you can simply print the contents of the file using the “more” command. E.g., more /etc/passwd
      2. Check this file to see if the users were created with the correct home directories, and use ls to confirm that those directories actually exist.
2. **Create a new group and add a user to that group**
   1. **Program to use:** groupadd.
      1. Hint: Run “man groupadd” to find information on how to add a group.
   2. Adding a user to a group: **Program to use:** “usermod”
3. **Lookup user and group id’s**

See the entries in the /etc/passwd file. The third and 4th records specify the user and group id’s respectively.

1. **Set the expiry dates on the accounts of the users you just created. By setting an expiry date you are protecting against dormant accounts.** 
   1. **Program to use:**  usermod
   2. **Sample usage:** usermod –e 2013-22-12 spock
   3. See “man usermod” on how to use this command.
2. **Setting and checking expiry date and warning dates on user passwords.** 
   1. First create a password.
      1. **Program to use:** chpasswd
      2. **Sample usage:** echo uhura:security1# | chpasswd
   2. Check a user’s password characteristics:
      1. **Program to use:** chage
      2. **Sample usage:** chage --list uhura (or, chage –l uhura)
   3. Change a user’s password expiry date:
      1. **Program to use:** chage
      2. **Sample usage:** chage –e 100(password expires in 100 days)
   4. Change a user’s password warning date: By default the warning is given 7 days before expiry date.
      1. **Program to use:** chage
      2. **Sample usage:** chage –W 100(warns 100 days before).
3. **Determining default permissions**
   1. As Spock, create a new file in Spock’s home directory.
      1. Change the current-user[[1]](#footnote-1) to spock. As root you can do this with: “su spock”.
      2. Go to Spock’s home directory (remember: cd ~)
      3. Use the command: “touch test.txt” to create test.txt file.
         1. Using ls, verify that the file test.txt was successfully created.
         2. What are the permissions with which the file was created?
            1. Use the ls –l test.txt command to see its permissions. Here’s a sample output of ls –l:

-rw-r—r-- 1 root root 0 2012-04-11 10:53 test.txt

* Here’s how to interpret these permissions:
  + Record 1: Permissions flags – ten of them.
    - First flag: indicates if it is a directory (d), a setuid program (s), or just a file (-).
    - Next three: Does the *owner* have read, write and execute permissions respectively  
      (“-“ means no; any letter means yes).
    - Next three: Does the *group* have read, write and execute permissions respectively.
    - Final three: Does the *world* have read, write and execute permissions respectively.
  + Record 2: Link count (Number of hard links to this file)
  + Record 3: owner of the file (in this example, root)
  + Record 4: group of the file (in this example, the second root)
  + Record 5: size, in bytes
  + Record 6-7: date of last modification
  + Record 8: name of the file.
  1. Permissions as bits: The permission record (record 1) can be viewed as ten bits; a value of 1 indicates that permission is set, and 0 indicates no permission. An example:
     + 1. If permissions are: -r--r--r--, we can also interpret this as the binary numeral 0 100 100 100, or in octal as 0444 (because [100]2 = [4]8; three bits translates directly to one octal digit).
       2. The OS interprets and allows us to manipulate permissions using numbers.
  2. Changing permission (chmod)
     1. To change a file’s permission, use chmod. You can specify relative or absolute permissions:
        1. Relative: You can add/remove existing permissions, leaving all other pemissions the same. For example, to add a **w**rite-permission for the file’s **g**roup, use: chmod g+w test.txt
        2. Absolute: You set all permissions (overwriting all previous bits) by specifying the bit-pattern as an octal numeral. To give, say, permissions -rw-r--r-- to a file, we compute that the bit pattern 0 110 100 100 is 0644 in octal, so use: chmod 644 test.txt

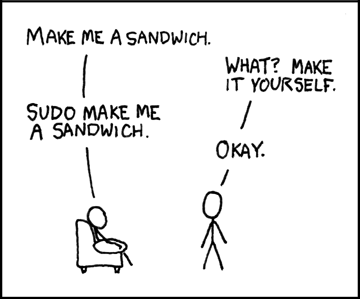
1. **Changing default permissions**: When a file is created it is assigned default permissions of 666. In addition, masking bits called *umask* are used to further modify the default permissions by masking out selected permission-bits. E.g., if umask is 022, then a new file’s permissions will be: (666 AND NOT 022) = (110 110 110 AND NOT 000 010 010) = (110 100 100) = 644.

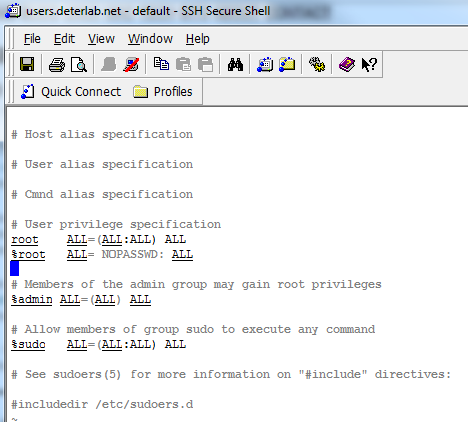
Each user has their own umask value, and can change it with with umask command. The initial value is set when a user logs in and runs the system-wide file /etc/login.defs .

Try it: Change users’ default file-permissions to be 444 by manipulating umask in the login.defs file. (Also, to test this set USERGROUPS\_ENAB to no in /etc/login.defs, since that setting further modifies a file’s group-permission in specific circumstances.) Re-login (or, su) as spock, create a file, and confirm it has the desired permissions.

Try it: Change test.txt’s group to “security”. You can do this by: “chgrp security test.txt”

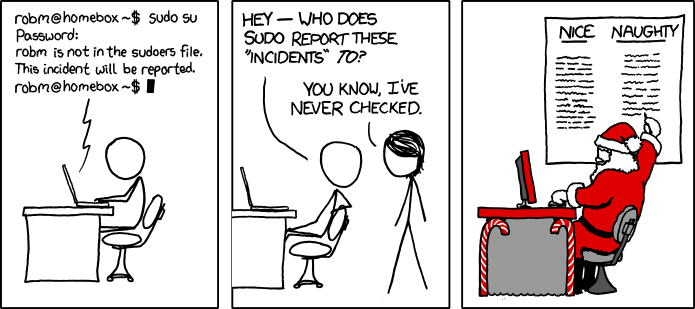
1. **Identify the processes that are running on your system**
   1. **Program to use:** ps.
   2. **Sample usage:** calling ps with options –aux prints all the running processes.
2. **Identifying processes that are using up most resources.**
   1. **Program to use**: top
3. **Checking for all services that are running on the system**.
   1. **Program to use:** chkconfig**.**
   2. **Note: this program may not be installed on your system; if not you can install with a package manager:** sudo apt-get install chkconfig
4. **Switch on a specific service:**
   1. **Program to use:** chkconfig
   2. Note: the service will not be automatically started up until the next reboot.
   3. Note: you may get the error “/sbin/insserv: No such file or directory”. If so, the program was not installed, but you’ll still get credit for the correct chkconfig command.
5. **Identifying, installing and removing programs on Linux (debian version)**
   1. **Program to use: apt-get.** Apt-get is a package manager. In Red hat Linux, the package manager is called yum. Package managers can be used to install, remove and check what packages are available. Use “man apt-get” to find various options.
6. **Giving sudo permission to users[[2]](#footnote-2)** 
   1. **Introduction**: We have seen that sudo allows a user to execute a program with root privileges. Clearly, you do not want everyone to have the ability to “sudo”. You can restrict not only who is able sudo at all, but you can also allow some users to only use sudo for certain commands.

© xkcd cartoon number 149

* 1. **Open the /etc/sudoers file.**
  2. **You will see entries such as: **
  3. The `root` line means that user root can execute from **ALL terminals**, and can execute as **ALL** users and can execute **ALL** programs.
  4. If you add an entry such as

spock ALL=/sbin/shutdown

then spock can execute /sbin/shutdown with root privileges, but spock cannot execute any other program with root privileges. This is one example of how to ensure that users have only restricted root privileges. If spock tries executing any *other* program via sudo, he will get an error message.

****

**© xkcd www.xkcd.com**

Assignment created by Dr. Prem Uppuluri, and modified by Ian Barland.

1. Note that changing-the-current-user (su) is not quite the same as doing a full login; the latter starts a sub-shell with its own environment variables, runs any commands in ~/.login, changes the directory to ~, *etc.* Also, note that login is a program you can run from the shell any time; it just happens to also be the default program by the server when you connect via ssh. [↑](#footnote-ref-1)
2. Notes from: <http://www.garron.me/linux/visudo-command-sudoers-file-sudo-default-editor.html> [↑](#footnote-ref-2)