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 see your permission being denied, ask yourselves whether this is because you don’t have root privileges. Use sudo to get root privileges.*

Due date: Nov.06 (Wed.) 14:00.

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.
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. First login as spock. From the root account, you can do this as follows: “su spock”.
		2. Go to Spock’s home directory (remember: cd ~)
		3. Use the command: “touch test.txt” to create test.txt file.
			1. Verify that the file test.txt was successfully created using the ls command.
			2. What are the default permissions with which the file was created?
				1. Use the ls –l test.txt command to find the default 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.
		- There are 10 permission flags as follows:
			* First flag: indicates if it is a directory (d) or a setuid program (s) or just a file (-).
			* Next three flags: Does the owner have read, write and execute permissions respectively.
			* Next three flags: Does the group have read, write and execute permissions respectively.
			* Final three flags: 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 the example root)
	+ Record 4: group of the file (in the 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 treated as if there were 10 binary bits. Bit 1 indicates that permission is set and 0 indicates no permission. Here is an example:
		+ 1. If permissions are: -r--r--r--, we can also interpret this as a binary number: 0100 100 100 or in groups of 3 starting from right as: 4 4 4 (because 100 in binary is 4 in decimal).
			2. The OS interprets and allows us to manipulate permissions using numbers.
	2. Changing permission (chmod)
		1. To change a permission, use chmod. E.g., to add a permission “write” to say the group, you can use the command: chmod g+w test.txt. Alternatively, you can change permission by specifying the permission number. Let us say we wish to give permissions –rw-r--r—we can do the same as 0110100100 which is 644 in octal as follows: chmod 644 test.txt
1. **Changing default permissions**: When a file is created it is assigned a default permission of 666. In addition, a masking bits called umask is provided in UNIX to change the default permission as follows. E.g., if umask is 022, then the permissions will be 666 XOR (Exclusive OR) 022.
	* + 1. umask is in the file /etc/login.defs by default.

Change the default permissions to 444 by manipulating umask in the login.defs file.

* 1. Change the group of the file 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:** Ps with the options –aux prints all the processes running on the system.
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. In which case first install it as follows:**
		1. **Run: 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. **~~and (13)~~ 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[[1]](#footnote-1)**
	1. **Introduction**: So far, 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”!. So you can restrict both who can “sudo” at all, as well as what commands they can “sudo”.

© 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. Now, if spock tries executing any other program with root sudo, he will see an error message.

****

**© xkcd www.xkcd.com**

1. Notes from: <http://www.garron.me/linux/visudo-command-sudoers-file-sudo-default-editor.html> [↑](#footnote-ref-1)