Utilizing sudo on z/OS Hands-on Lab

SHARE in Anaheim
Session: 11636 on August 7, 2012

C.T. Ware
IBM Poughkeepsie, NY
ctware@us.ibm.com

Trademarks and Disclaimers

See http://www.ibm.com/legal/copytrade.shtml for a list of IBM trademarks.
The following are trademarks or registered trademarks of other companies:

- UNIX is a registered trademark of The Open Group in the United States and other countries
- CERT® is a registered trademark and service mark of Carnegie Mellon University.
- ssh® is a registered trademark of SSH Communications Security Corp
- X Window System is a trademark of X Consortium, Inc

All other products may be trademarks or registered trademarks of their respective companies

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices are subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Lab Overview

Objectives of this lab

At the end of this lab, you will be able to do the following:
- Utilize sudo.

This lab is written for IBM Ported Tools for z/OS: sudo utility. Refer to the "IBM Ported Tools for z/OS: Supplementary Toolkit for z/OS Feature User's Guide and Reference" for more information.

Note: The sudo –V option can be used to verify the sudo release in use. The IBM port output should be "`Sudo version 1.7.2p2`".

How is this lab different than the real world?

1. You do not have system administrator privileges. So...
    - You will not have the ability to edit /etc/sudoers (the sudo config file)
        - This has been configured ahead of time to allow you to run sudo with only the 'chown' shell utility; in the real world this could be configured to allow any designated ID access to any designated command(s).

The following example information box will help identify when this lab is different than the real world.

Information for the real world…

- This is the information for the real world…

Typing suggestions to get through this lab

If you prefer to copy/paste commands, a text file in your $HOME/ptoolslab directory contains the commands for this lab. You can view it using "more" or "cat". For example:

```
prompt=> cat $HOME/ptoolslab/sudocommands
```

> **Note:** This file is personalized for your SHARE user ID. You may want to login twice (opening 2 PuTTY sessions) if using this file so you can copy the lab commands from one session while doing the lab exercises in another session. By default, the right mouse button can be used to copy highlighted text in a PuTTY session and the left mouse button can be used to paste that text.

Another alternative is to use the /bin/tcsh shell to take advantage of the up and down arrows for retrieving previous commands. However, this lab is designed for use with the /bin/sh shell. It is recommended that you don't use the /bin/tcsh shell unless you are familiar with the /bin/tcsh versus /bin/sh shell differences (e.g. syntax for command substitution, environment variable specification, etc.).

Other important lab information

This SHARE lab is self-paced and self-contained. However, you need to go in order; don't do the lab exercises out-of-order. Also be sure to **substitute your SHARE user ID** when the lab shows **shar___** or **SHAR___** from now on (e.g. sharb04 where 'b' is the lab letter and '04' is your lab number).

Your SHARE user ID went through some basic configuration in preparation for this lab. The configuration consisted of miscellaneous file (e.g. $HOME/ptoolslab/sudocommands), and directory (e.g. $HOME/ptoolslab) and setup. If you would like more information on this configuration or on the general setup for this lab, please contact the lab presenter.

This document shows the /bin/sh shell prompt like the following. Note that that actual text of your shell prompt will vary.

```
prompt=>
```

This document shows the commands to be run after the shell/command prompt like the following:

```
prompt=> cat $HOME/ptoolslab/sudocommands
```

Some of the commands run are very long and may get truncated on the display like the following:

```
prompt=> udocommands  <
```

If this occurs, do the following to increase the number of columns displayed:

```
prompt=> stty columns [number of columns]
```

This should already be configured for you, but if while in your PuTTY session, you press the backspace key and control characters are displayed (i.e. it doesn't actually delete the previous character), for example:

```
prompt=> ^H ^H
```

Do the following to map your backspace key:

```
prompt=> stty erase [press Backspace key]
```

In the real world, this command can be added to a shell profile (/etc/profile or user-specific $HOME/.profile), or your PuTTY configuration can be updated to pass the proper backspace key

Lab Exercises

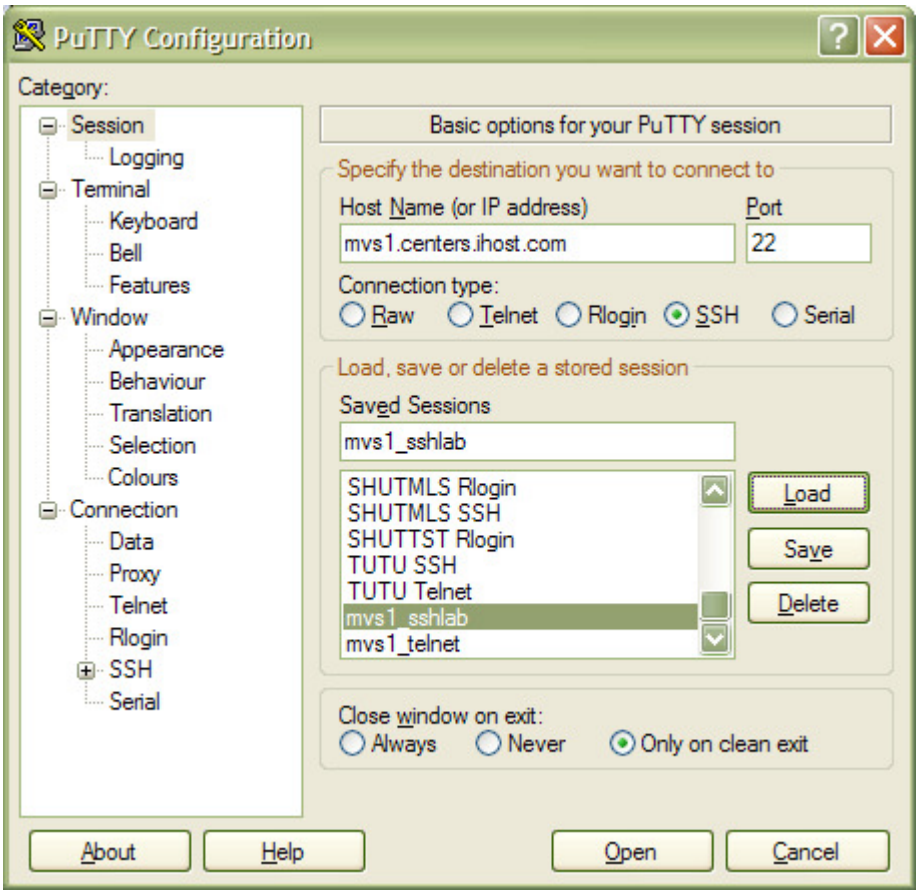
Where do I start?

☐ Login to the z/OS host

How to logon using PuTTY:

☐ Start a PuTTY session. Double-click on PuTTY.

The following configuration window will appear:



- ☐ Load the "mvs1_sshlab" saved session. If the session does not exist, enter the following fields:

```
Host Name:                mvs1.centers.ihost.com
Port:                     22
Connection Type:          SSH
```

Note: If you don't have PuTTY on your workstation, refer to the Appendix on where to download it. You can also telnet into `mvs1.centers.ihost.com` using port `623`.

- ☐ Click “Open”.

Note: If you are using SSH to connect to an sshd daemon/server for the first time, you may see a message looking something like the following:

```
The server's host key is not cached in the registry. You have no
guarantee that the server is the computer you think it is.  The
server's key fingerprint is:
ssh-rsa 1024 7b:e5:6f:a7:f4:f9:81:62:5c:e3:1f:bf:8b:57:6c:5a
If you trust this host, hit Yes to add the key to PuTTY's cache and
carry on connecting.  If you want to carry on connecting just once,
without adding the key to the cache, hit No.  If you do not trust
this host, hit Cancel to abandon the connection.
```

This is a feature of the SSH protocol. It is designed to protect you against a network attack known as spoofing: *secretly redirecting your connection to a different computer, so that you send your password to the wrong system. Using this technique, an attacker would be able to learn the password that guards your login account, and could then log in as if they were you and use the account for their own purposes.*

For now, hit yes to add the key.

- ☐ Enter your SHARE user ID and password. Remember to **substitute your SHARE user ID** when the lab shows **shar___** or **SHAR___** from now on (e.g. sharb04 where 'b' is the lab letter and '04' is your lab number).
  - User ID: **shar___**
  - Password: **[The lab presenter will provide the password.]**

Configuring the sudo utility…

☐ The files for sudo are located in /usr/lpp/ported and should have been linked to /usr/bin as part of the component installation. (along with the respective links for the man pages and the sample sudoers file copied to /etc/sudoers). To verify sudo is linked properly, you can invoke sudo with the –V option to return the sudo version:

```
prompt=> sudo -V
   Sudo version 1.7.2p2
```

The /etc/sudoers file contains all the configuration data for sudo. You must be UID 0 to run visudo, which is the recommended method of updating this config file (as it does some error checking of the file). To run visudo, it is recommended you use PuTTY, but if you're unable to (or more comfortable with tn3270) you can use your favorite MVS editor to alter the file (but the error/syntax checking will not be done, and sudo will not run if the /etc/sudoers file has errors).

To demonstrate the security of /etc/sudoers, let's try to run visudo:

```
prompt=> /usr/sbin/visudo
   /usr/sbin/visudo: FSUM9209 cannot execute: reason code = 5b4c0002:
   EDC5111I Permission denied.
```

You will notice as a non-UID 0 user, you cannot run visudo. For another attempt to "access" /etc/sudoers, we can try the 'cat' shell utility (which would dump the contents of a file to the terminal):

```
prompt=> cat /etc/sudoers
   cat: /etc/sudoers: EDC5111I Permission denied.
```

Once again, /etc/sudoers is protected; so the ability to edit and view /etc/sudoers is restricted to UID 0 users.

Information for the real world…

- If you were UID 0 and configuring /etc/sudoers using visudo, you would have the ability to grant any ID privileged access to any command.

Using the sudo utility…

☐ The sudo utility takes as input the shell command you wish to execute with superuser authority. These steps assume your ID has been granted /bin/chown access by the lab presenter in the /etc/sudoers file and will demonstrate the abilities of sudo. First we need to set up a scenario you'll need superuser access for. We're going to create a new file, then verify its setup. Following this, we're going to try and change its owner to UID 0, which we should expect to fail; however, using sudo, we should be able to successfully change its owner. We'll then verify and try to delete the file (which *may* fail); once again we'll use sudo to change its owner back to our ID and then successfully remove the file. Here we go, remember to replace the __ with your appropriate value (i.e. your file should be similar to sharb02.myfile – all lowercase):

```
prompt=> touch $HOME/ptoolslab/shar__.myfile
prompt=> ls -al $HOME/ptoolslab/shar__.myfile
   -rw-r--r--   1 SHAR__   1              0 Jul 18 14:59 shar__.myfile
```

The first 'touch' command simply creates an empty file, and in the output of the second command 'ls' tells us information about the file. Where you see 'SHAR__' above, in your output should be your SHAR__ ID, this verifies the file was created and is owned by you.

☐ Next we're going to attempt to change the file owner to UID 0, which should fail:

```
prompt=> chown 0 $HOME/ptoolslab/shar__.myfile
   chown: FSUM6180 file "$HOME/ptoolslab/shar__.myfile": EDC5139I
   Operation not permitted.
```

☐ If the above failure occurs you've discovered you don't have the permissions necessary to change the owner of your file to UID 0. So let's use sudo to gain superuser privileges and perform the chown successfully:

```
prompt=> sudo /bin/chown 0 $HOME/ptoolslab/shar__.myfile
```

Note: when running sudo, it will prompt for a password, this is **your** IDs password – not a superuser password.

If you're *not* prompted for your password it is because sudo will keep your authority valid for a limited amount of time (default is 5 minutes, but can be configured in the /etc/sudoers file under the 'passwd_timeout' option).

☐ Now verify the 'chown' was successful using the 'ls' output (similar to before) and check the current file owner, it should be a UID 0:

```
prompt=> ls -aln $HOME/ptoolslab/shar__.myfile
   -rw-r--r--   1 0        1              0 Jul 18 14:59 shar__.myfile
```

Note: by including the 'n' option on the 'ls', the output will show the UID number, instead of the username; and should be zero.

☐ Next try to delete the file you've created, and you *may* encounter a permission denied failure; after all, you no longer own the file:

```
prompt=> rm $HOME/ptoolslab/shar__.myfile
   rm: FSUM9195 cannot unlink entry "$HOME/ptoolslab/shar__.myfile":
   EDC5111I Permission denied.
```

Note: This fails if the sticky bit is set, on the owning directory.

☐ So to remove this file, we're going to have to change the owner back to our SHARE ID, but we'll need superuser privileges to do so:

```
prompt=> sudo /bin/chown shar__ $HOME/ptoolslab/shar__.myfile
```

☐ And now you should be able to remove the file without any problems:

```
prompt=> rm $HOME/ptoolslab/shar__.myfile
```

Still curious, here are a few more things to try…

If you have extra time and want to try a few things, here are some other options that you may want to try.

☐ Issue the man command for more information on the sudo command.

```
prompt=> man sudo
```

If this doesn't work, the MANPATH has likely not been setup properly, the following should give usable results:

```
prompt=> man -M /usr/lpp/ported/man/%L sudo
```

sudo options to consider trying:

-k	→ (kill timestamp) Removes sudo's timestamp for this user, will force the next sudo command to reprompt for a password.
-k command	→ Removes sudo's timestamp for this command, will force prompting for your password.
-n	→ Forces sudo to run in non-interactive mode, if the user's timestamp has expired, it will not prompt for a password and will fail.

☐ Consider trying to create a script called 'chown' to fool sudo and perform something requiring UID 0 access. Here's an example, this is the content of $HOME/ptoolslab/chown (which has been created for you already):

```
#!/bin/sh
touch /etc/rootfile
```

This script will try to create a file in the /etc/ filesystem, which is restricted (if you invoke this command directly you should receive the following error:
`touch: file "/etc/rootfile": EDC5111I Permission denied.`
So this tries to circumvent this using the knowledge of having superuser access to /bin/chown, by creating an executable script with the same name (different directory).

You could also try the following:

```
prompt=> sudo /bin/touch /etc/rootfile
```

which will fail (you haven't been given access to run 'touch' under sudo).

So what do you think will happen if you run:

```
prompt=> sudo $HOME/ptoolslab/chown
```

experiment and find out!

This is the end of the lab.
Hope you had fun!

Appendix

Shell command-line editing quick reference

Issue the following to enable vi command-line editing: `set -o vi`
Note: This has already been enabled by default on the SHARE system.

To leave **insertion** mode and enter **command** mode (so the characters you type are understood as commands), press the Escape key [ESC]. Do this before using the commands below. While in **command** mode, [ESC] will return you to **insertion** mode.

To do the following (after [ESC]):	Type this command:
Recall previous command line	k
Move cursor left	h
Move cursor right	l (this is a lowercase L)
Insert characters after cursor	i
Append characters after cursor	a
Replace characters	R[type your text][ESC]
Replace 1 character	r[type your character]
Delete 1 character	x
Execute command line (while in command mode)	Enter (when line is displayed)
Discard command line	^C (Ctrl-C)
Complete filename	\

Documentation

IBM Publications for IBM Ported Tools for z/OS:
http://www.ibm.com/systems/z/os/zos/features/unix/ported/

sudo Home Page:
http://www.sudo.ws/

PuTTY download

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

How to logon to TSO (3270 interface)

Start the emulator

1. Double click on **SHARE System** icon. This starts a PCOMM 3270 session using mvs1.centers.ihost.com. **Note:** The Enter key is the right Ctrl key
2. You can skip this step for now, but for real use, you may want to configure session parameters to use:
   Screen Size: 43x80
   Host Code Page: 1047 United States

Logon to TSO/E

1. When prompted for Userid/Password/Application, enter TSO in the Application field and press the Enter key.
2. User ID: **shar___**
3. Password: **[The lab presenter will provide the password.]**
4. ISPF will be started
5. From ISPF, enter option 6
6. Enter: `omvs esc('@')`
   This starts a login shell with an escape character of '@'. The escape character is used to simulate the Ctrl key. The default is the cent sign, which would need to be configured in the emulator. You can also configure the emulator so that popular Ctrl keys (e.g. Ctrl-C, Ctrl-Z) generate the appropriate OMVS escape sequence. With the above command, to interrupt a running command, you enter @c on the command line.

For example, if you see:

```
[press Ctrl-D]
```

You will instead:

```
[press @D Enter]
```

The Enter key is required because a 3270 session is a line mode terminal.