Linux Lab 8 Installing and Removing Software (Ubuntu)

Read

Listen to or read CyberAces Module 1 Linux, session 7 Installing Software (https://tutorials.cyberaces.org/tutorials/view/1-1-7.html)

The Linux Command Line, chapter 14 Package Management and chapter 23 Installing Software, pp 149 - 157 and 297 - 306 in the printed book and pp 172 -18 and 350 - 361 in the pdf.

These are some good references for how the standard "./configure , make , make install" , sequence works.
http://tldp.org/LDP/LG/current/smith.html (see paragraph 6 for an old Linux joke)
http://www.opensourceforu.com/2012/06/gnu-make-in-detail-for-beginners/

Lab

First, we will install nmap (a network mapping or discovery application, https://nmap.org/) using software repositories and the apt commands. Then we will remove nmap, and install it from source code to practice using "./configure, make, and make install" to compile and install software.

The main advantages to installing from a repository are that it is easy, and that your application will be automatically updated from the repository in the normal `apt-get update; apt-get upgrade` cycle. However, the repository does not always have the most current version of the application. If you need a version that is not in the repository, or one that is specially compiled you will have to install from source code. If you do install from source code, you will have to update the application manually.

1) Is nmap in our repository?

Sometimes the name is slightly different from what we think it is, or the application is not in the repository. Page 152 of *The Linux Command Line* shows us how to check. (Note: the command

```
apt update
```

will update the VM's database of available software, and is run before doing anything with apt.) You will get multiple answers--you are looking for one that starts with 'nmap'.

```
sudo apt update
apt search nmap
```

Note: You can also Google something like "nmap install ubuntu" to find the name that nmap goes by in apt.

2) Check to see if nmap is installed.

From the text, page 155, we can use dpkg (Debian Package Manager) to see if nmap is installed. On my Ubuntu 18 VM, it was not installed.

`dpkg --status nmap` or `dpkg --list | grep nmap`

or we can use apt

```
apt list --installed nmap
```

Repositories (you can think of them as app stores) are the easiest way to install software, but you're relying on someone to put the program you need into the repository, and it may not be the most

current version. However, if the version supplied in the repository meets your needs and is kept up to date, your best choice is to install using `apt` and the repository. That way, you can update all your software by just typing `apt update` and `apt upgrade`. If you install from source code, you'll have to apply patches yourself.

Nmap also kindly supplies precompiled versions using RPMs, which are packages built for Redhat. You can see simple instructions for using these at https://nmap.org/download.html, in the section "Linux RPM Source and Binaries." They don't provide Debian Package Files (.debian) that we can use with Ubuntu, though.

3) Install nmap from the repository. Once you know the package name, it's easy.

```
sudo apt install nmap
```

When you've finished, the status command we used before should show that it is installed.

```
dpkg --status nmap
```

Run `sudo nmap -h`. What version of nmap do you have installed?

4) Remove nmap

Note: Installing nmap from source code in Ubuntu 18 is messy at the time this is written. If you like, you can skip removing nmap and reinstalling it from source code. If so, read the paragraph about `tar` and then skip ahead to review. If you are an intrepid Linux user, go ahead and remove nmap and install from source code.

```
sudo apt remove nmap
```

5) Install nmap from the source code. (Optional)

When you install from source code you can make sure you have the latest version, or the version you want. Remember that if you install from source code, `apt upgrade` won't help you; you'll have to install updates manually.

Since we are compiling from source code, we will need to install a compiler and related development software. This command does that quickly. (You may find that it is already installed.)

```
sudo apt install build-essential
```

Go to https://nmap.org/download.html and download the latest tarball of nmap—it's at the bottom of the section called "Source Code Distribution," labeled "Latest Stable Nmap release tarball." The tar, or tape archive program, is used to put multiple files into one file and optionally compress them. (Tar (tape archive) files are often called tarballs. Tar was originally used for backup--the files on the computer were put into a tarball (a single file, like zip but without compression) and the tarball was then saved to magnetic tape.) Nmap now compresses its files with bzip2 rather than gzip.

Save the file rather than opening it with the web browser. If you are a hard core terminal person, you can download using curl or wget. You still must know the download URL, though. You can often get the link using right-click "copy link location" in the browser, and then pasting it. As I write this, the link is https://nmap.org/dist/nmap-7.92.tar.bz2.

If you download the file using the browser in the VM, it will be in the ~/Downloads directory.

Extract the nmap files using the tar program. You'll need to use the options to extract (`-x`) and uncompress (use `-j` for bzip2 or `-z` for gzip files, depending on which one you downloaded) the files. It's nice to use the verbose option (`-v`) so you can see what's going on, and you'll need to use an option (`-f`) to tell tar that the data is coming from a file (and what its name is.) It doesn't matter much what directory you untar the file to. I chose to move the tarball to my home directory before uncompressing it, although a neater user will create a directory for source files.

`tar -xjvf nmap-[version].tar.bz2` (bzip2 compressed) or
`tar -xzvf nmap-[version].tgz` (gzip compressed)

Note: [version] will be different, depending on what you download

```
john@ubuntu20f21:~$ cd Downloads/
john@ubuntu20f21:~/Downloads$ ls
nmap-7.92.tar.bz2
john@ubuntu20f21:~/Downloads$ tar -xjvf nmap-7.92.tar.bz2
nmap-7.92/
nmap-7.92/missing
nmap-7.92/lpeg.c
nmap-7.92/configure
nmap-7.92/tests/
nmap-7.92/tests/nmap_dns_test.cc
nmap-7.92/nmap_tty.cc
<snip>  lots of stuff omitted.
```

Change directory (cd) into the directory with the files you just unzipped. For most installations, there is a configure script that does the work of figuring out what distribution you have, where your compilers and libraries are, and it puts that information into something called the Makefile. Look for README and other info files (usually the file names are all Caps.) Sometimes there are special configuration options you can use, and these are often documented in the beginning of the configure file (`less configure`.) Look to see that there is a file called "configure" in your directory and scan the beginning with less to see if there are interesting options. Also look to see if there is a Makefile. (If configure hasn't run yet, should will be a Makefile.in but no Makefile.)

```
john@ubuntu20f21:~/Downloads$ ls -l
total 10260
drwxr-xr-x 24 john john     4096 Aug  7 11:37 nmap-7.92
-rw-rw-r--  1 john john 10498200 Sep  7 09:45 nmap-7.92.tar.bz2
john@ubuntu20f21:~/Downloads$ cd nmap-7.92/
john@ubuntu20f21:~/Downloads/nmap-7.92$ ls
acinclude.m4          libnetutil          nmap_error.h          nse_libssh2.h     output.h              string_pool.h
aclocal.m4            libpcap             nmap_ftp.cc           nse_lpeg.cc       payload.cc            struct_ip.h
BSDmakefile           libpcre             nmap_ftp.h            nse_lpeg.h        payload.h             Target.cc
CHANGELOG             libssh2             nmap.h                nse_lua.h         portlist.cc           TargetGroup.cc
charpool.cc           libz                nmap-mac-prefixes     nse_main.cc       portlist.h            TargetGroup.h
charpool.h            LICENSE             NmapOps.cc            nse_main.h        portreasons.cc        Target.h
checklibs.sh          lpeg.c              NmapOps.h             nse_main.lua      portreasons.h         targets.cc
config.guess          ltmain.sh           nmap-os-db            nse_nmaplib.cc    probespec.h           targets.h
config.sub            MACLookup.cc        NmapOutputTable.cc    nse_nmaplib.h     protocols.cc          tcpip.cc
configure             MACLookup.h         NmapOutputTable.h     nse_nsock.cc      protocols.h           tcpip.h
configure.ac          macosx              nmap-payloads         nse_nsock.h       README.md             tests
CONTRIBUTING.md       main.cc             nmap-protocols        nse_openssl.cc    README-WIN32          timing.cc
depcomp               Makefile.in         nmap-rpc              nse_openssl.h     scan_engine.cc        timing.h
docs                  missing             nmap-service-probes   nse_pcrelib.cc    scan_engine_connect.cc todo
FingerPrintResults.cc mswin32             nmap-services         nse_pcrelib.h     scan_engine_connect.h traceroute.cc
FingerPrintResults.h  nbase               nmap_tty.cc           nse_ssl_cert.cc   scan_engine.h         traceroute.h
FPEngine.cc           ncat                nmap_tty.h            nse_ssl_cert.h    scan_engine_raw.cc    utils.cc
FPEngine.h            ndiff               nmap_winconfig.h      nse_utility.cc    scan_engine_raw.h     utils.h
FPModel.cc            NewTargets.cc       nping                 nse_utility.h     scan_lists.cc         xml.cc
FPModel.h             NewTargets.h        nse_debug.cc          nse_zlib.cc       scan_lists.h          xml.h
HACKING               nmap-7.92-1.spec    nse_debug.h           nse_zlib.h        scripts               zenmap
idle_scan.cc          nmap_amigaos.h      nse_dnet.cc           nsock             service_scan.cc       zenmap-7.92-1.spe
idle_scan.h           nmap.cc             nse_dnet.h            osscan2.cc        service_scan.h
INSTALL               nmap_config.h.in    nse_fs.cc             osscan2.h         services.cc
libdnet-stripped      nmap_dns.cc         nse_fs.h              osscan.cc         services.h
liblinear             nmap_dns.h          nselib                osscan.h          shtool
liblua                nmap_error.cc       nse_libssh2.cc        output.cc         string_pool.cc
```

Configure will check to see what your operating system is and what libraries are available. It will put that information into a Makefile.

Configure

Too bad, no interesting options on this configure file. Run the configure script.

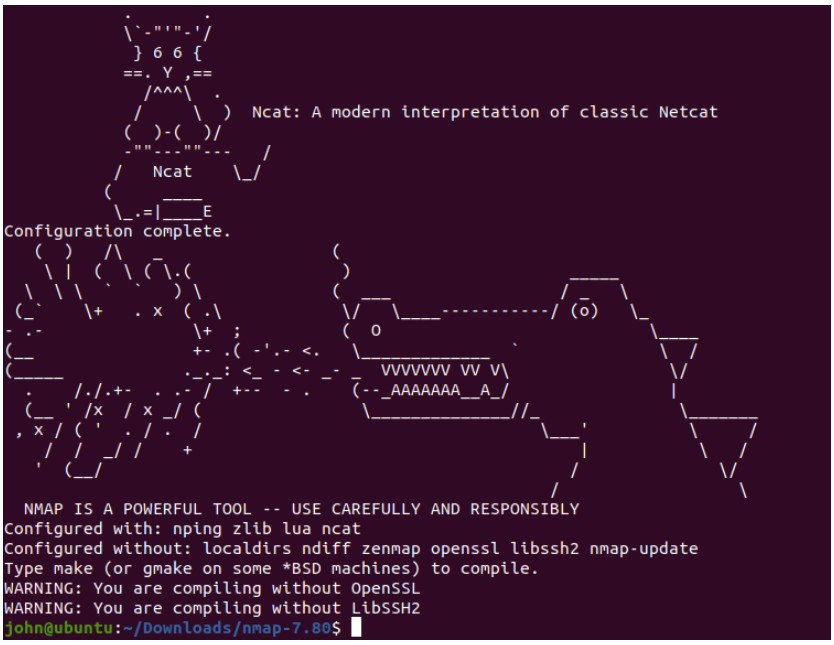
```
./configure
```

If you get a "command not found" error, remember what you need to do to run a script when you are in the same directory as the script. (Hint: the abbreviation for your current directory is ./) Also, if the configure script complains about not having a C compiler, you probably forgot to run `sudo apt install build-essential`.

Missing Dependencies

The biggest problem installing from source code is that the code often depends on other applications that are not installed on your VM. If there are missing dependencies, `./configure` or `make` will generate errors. Sometimes you must Google the error messages to find what is missing.

With Ubuntu version 20-04.1 and Nmap 7.92, we do not get errors for missing dependencies. Nice.



It tells us that we don't have OpenSSL and LibSSH2 installed, but that will only affect us if we try to decode SSL or SSH traffic. Now for `make`.

make

The configure script ran many checks, and if they all passed, generated a Makefile. Use `ls` to look for it. When you did this before, you should have seen only Makefile.in. Now you should see Makefile.in and Makefile. The command `make` calls a program that is already installed in your OS. If you run, "`which make`", you'll see that it lives in /usr/bin/. If it were missing, you'd have to run, "`apt install make`". `Make` saves you the trouble of looking for modules your software depends on (dependencies), compiling each module of the software, putting them in the proper order, and then linking them all together. For complex software, this is a lot of work. All you need to do is type one word:

`make`

You must type `make` from the directory that holds your Makefile.

**Why do you type `make` instead of `./make` ?

It will compile ~~about a gazillion~~ many files. If you run `ls`, you will see ~~a whole bunch of~~ many files that weren't there before. Most of these are *.o files, or object files. They are compiled binary files that can be linked together to run programs.

```
john@ubuntu:~/nmap-7.80$ make
Compiling libnetutil
cd libnetutil && make
make[1]: Entering directory '/home/john/nmap-7.80/libnetutil'
g++ -c -I../liblinear -I../liblua -I../libdnet-stripped/include -I../libz -I../l
ibpcre  -I../nbase -I../nsock/include -DHAVE_CONFIG_H -D_FORTIFY_SOURCE=2 -g -O2
 -Wall -fno-strict-aliasing netutil.cc -o netutil.o
```

<lots of output deleted>

```
gcc -o ncat -g -O2 -Wall   ncat_main.o ncat_connect.o ncat_core.o ncat_posix.o n
cat_listen.o ncat_proxy.o ncat_ssl.o base64.o http.o util.o sys_wrap.o ncat_lua.
o ../nsock/src/libnsock.a ../nbase/libnbase.a  -lpcap ./../liblua/liblua.a -lm -
ldl
make[1]: Leaving directory '/home/john/nmap-7.80/ncat'
john@ubuntu:~/nmap-7.80$
```

No errors. Whew!

sudo make install

When you run `make install`, you are telling `make` that you've already compiled the program (see the previous step) and now you want to copy all those object files and binaries into the necessary places on the OS, often to /usr/bin and /usr/local/bin. Those directories require root access, so you'll have to run as root or use sudo.

```
john@ubuntu:~/nmap-7.80$ sudo make install
Compiling libnetutil
cd libnetutil && make
make[1]: Entering directory '/home/john/nmap-7.80/libnetutil'
```

<snip>

```
/usr/bin/install -c -c -m 644 docs/nping.1 /usr/local/share/man/man1/
NPING SUCCESSFULLY INSTALLED
make[1]: Leaving directory '/home/john/nmap-7.80/nping'
NMAP SUCCESSFULLY INSTALLED
john@ubuntu:~/nmap-7.80$
```

Whew!

It works!

```
john@ubuntu:~/Downloads/nmap-7.80$ nmap
Nmap 7.80 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
```

Review

The basic commands we use to install from source code were:

- `tar`, to decompress the files
- `./configure` from the directory tar made, to check for dependencies and create the makefile
- `make` (same directory as above), to compile all the code
- `sudo make install` (same directory) to move the compiled files to the proper directories

Hand in

1) Were you able to install nmap using both methods? (It's ok if the source code method didn't work, it is not easy.)

2) What is nmap, anyway?

3) What are possible problems with installing programs from source code?