# Linux Lab 10 Installing and Removing Software (CentOS)

# Read

CyberAces\_Module1-Linux\_7\_InstallingSoftware.pdf (<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

Install nmap from the CentOS repository using the procedure in slides 11 through 16 in CyberAces\_Module1-Linux\_7\_InstallingSoftware.pdf.

Then update your CentOS VM by following slides 20 and 21 in CyberAces\_Module1-Linux\_7\_InstallingSoftware.pdf.

## 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, sudo yum update will not help you; you will 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.  
yum groupinstall "Development Tools"

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 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](http://www.compciv.org/recipes/cli/downloading-with-curl/) or [wget](https://www.pair.com/support/kb/paircloud-downloading-files-with-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-7.80.tar.bz2>.

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@localhost Downloads]$ ls

nmap-7.91 nmap-7.91.tar.bz2

[john@localhost Downloads]$ cd nmap-7.91/

[john@localhost nmap-7.91]$

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, checks whether you have all the software the new application depends on, 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@localhost Downloads]$ cd nmap-7.91/

[john@localhost nmap-7.91]$ ls

acinclude.m4 nmap\_error.cc osscan.h

aclocal.m4 nmap\_error.h output.cc

BSDmakefile nmap\_ftp.cc output.h

CHANGELOG nmap\_ftp.h payload.cc

charpool.cc nmap.h payload.h

charpool.h nmap-mac-prefixes portlist.cc

checklibs.sh NmapOps.cc portlist.h

config.guess NmapOps.h portreasons.cc

config.sub nmap-os-db portreasons.h

configure NmapOutputTable.cc probespec.h

configure.ac NmapOutputTable.h protocols.cc

<snip>

### Configure

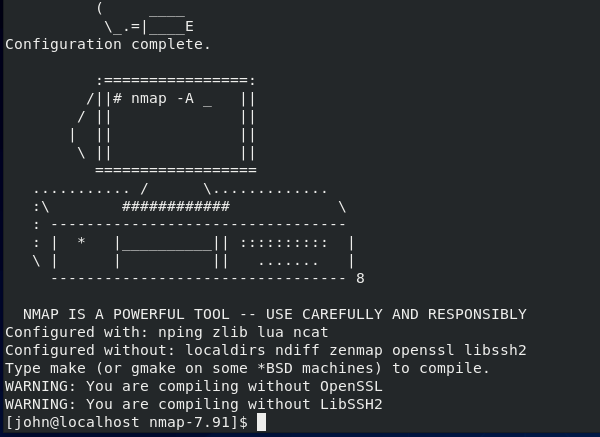
Too bad, no interesting options on this configure file. Run the configure script.  
./configure

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  
yum groupinstall "Development Tools"

### 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 have to Google the error messages to find what is missing.

With CentOS 8.2 and nmap 7.91, we do not get errors for missing dependencies. Nice.

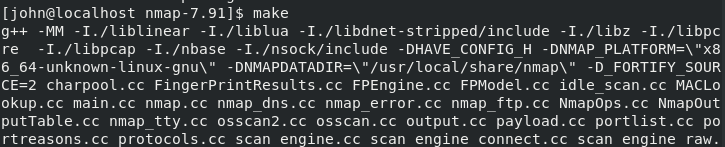
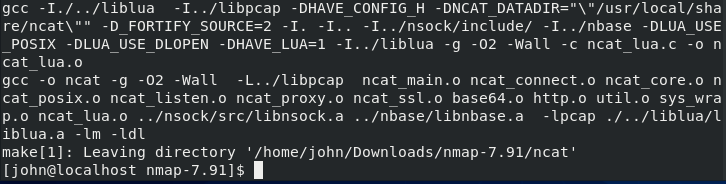


It tells us that we don’t have OpenSSL and LibSSH2 installed, but that will only affect us if we try to decode SSL or SSH traffic. Now for make.

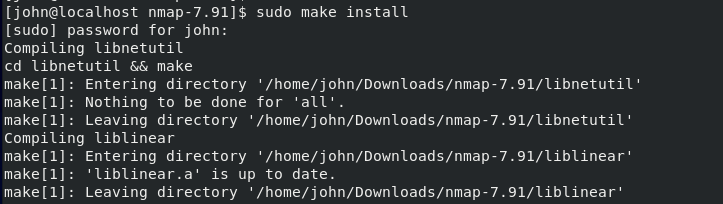
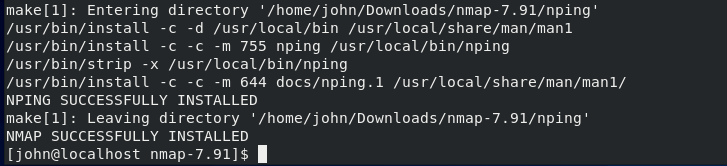
### make

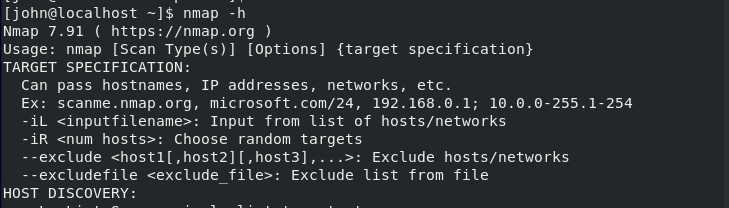
The configure script ran many checks, and if they all passed, generated a Makefile. Use ls to look for it. When you did this before, you should have seen only Makefile.in. Now you should see Makefile.in and Makefile. The command make calls a program that is already installed in your OS. If you run, “which make”, you'll see that it lives in /usr/bin/. If it were missing, you’d have to run, “apt-get install make”. Make saves you the trouble of looking for modules your software depends on (dependencies), compiling each module of the software, compiling 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'll 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.

  
<lots of output deleted>  
  
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.  
  
<snip>  
  
Whew!

Check to see if it runs.  
  
<snip>

Looks good!

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?
2. What is nmap, anyway?
3. What are possible problems with installing programs from source code?