To Dot or Not To Dot

William H. Mitchell

Last Revised: January 24, 2005 at 10:18pm

Overview

In short, having the current directory (.) in your search path can lead to your account being broken into. However, if the current directory is not in your path, executables in the current directory must be prefixed with "./" to be run—a significant inconvenience. In many cases, a reasonable balance between convenience and security can be achieved by putting the current directory at the end of the path.

Discussion

If the current directory is at the start of the path then when a command is typed, the shell first looks in the current directory for an executable with that name. If found, it runs it. Consider this command:

```
$ ls
a b c
```

What just happened? It looks like the user ran the ls command and it listed three files in the current directory. No harm done, right? Wrong—maybe.

Let's imagine the user had the current directory at the start of his path. Further imagine that in the current directory there was a script named ls, with these contents:

```
rm ./ls
ls $*
chmod 777 $HOME
```

Because the current directory was first in the user's path, ./ls was executed instead of /usr/local/bin/ls or maybe /bin/ls. The first thing the script does is to delete itself. (The shell already has it loaded, so that doesn't stop execution.) The script then executes ls and because ./ls is gone, the user sees the expected output. The script then changes the permissions on the user's home directory to mode 777, **giving every user full access to this user's home directory**—sort of like leaving your car unlocked in a bad neighborhood.

"Where did this malicious replacement for ls come from?", you ask. That depends on where you are. /tmp is a directory that some programs use to create temporary files. *Any* user can create a file in /tmp. A script like ls shown above could be put in /tmp. With that in place, the next user who has the current directory at the front of their path and who does this, for example,

```
$ cd /tmp
$ ls
```

has unwittingly opened up their home directory.

Maybe you're thinking, "OK, I see that it's dangerous to be in /tmp if the current directory is at the start of my path. If I stay out of /tmp is there any risk in having dot first?" Yes, there's risk in being in any directory that's "world-writable". /tmp is one example, /var/tmp is another. Through accident or intention, other world-writable directories may exist on a system. Each one is a dangerous place.

Risk is not limited to world-writable directories. A malicious user might leave their home directory accessible (a "honeypot") and put an ls replacement there. A snooper with dot at the start of the path would fall prey to it. Another example: a desperate student might turnin an ls replacement, hoping an unwary instructor or TA would cd to the directory with the submission and have a look with ls.

With all that in mind one might say, "I'll put the current directory at the end of my path. That'll make sure that the standard versions of programs are the first to be found by the shell. A malicious version of ls or who would never be reached. That's safe, right?" Having dot at the end of the path is safer but there's still some exposure: A malicious user might put in place a script that's a mistyped command name, such as sl or hwo. That requires more coincidence of course—a user would have to be in an untrusted directory that contains a malicious, misnamed script and then commit the anticipated error.

One might then say, "There's always some risk in having the current directory in your path, right?" That is correct. Thinking further, "Why allow this risk at all—wouldn't it make sense for shells to simply prohibit having the current directory in the path?" Risk is one factor in the equation but the other is convenience. If everyone is the user community is trusted—maybe it's a start-up and everybody has root (administrator) privileges—then it's probably a waste of time to leave dot out of your path and always type *./whatever*.

If you're an ordinary user on a machine (such as a student on an academic computing system) and can be mindful of the risk of issuing commands in untrusted directories, then having the current directory at the end of your path is a reasonable compromise between risk and convenience. Including the current directory in your prompt (PS1="\w $ " in bash) is a good way to remind yourself of where you are. (A dynamic prompt using command substitution opens up a lot of possibilities. The author has not investigated a third idea, dynamically altering the path based on the current directory, but that could provide a real solution, such as having dot in your path only when in your home tree.)

You might be wondering who would take the risk of doing something like putting a malicious ls or in /tmp. If the script was spotted before it was executed then '/bin/ls -l /tmp/ls' would show who created it, so it would be pretty stupid to try that, right? Wrong! A clever criminal would first steal an account—perhaps by spying a password being typed—and then, in the guise of that unwitting user, create the malicious /tmp/ls. If it is detected before triggered, then the user whose password was stolen would be first person to be questioned. (Another good reason to guard your password!)

More discussion of this issue can be found by Googling for "current directory in your path", "dot in the path", etc.

The attack shown above—"opening up" a user's home directory—is one of many possible first steps. A more common route is to create a SUID (set user id) script.

Life would sure be a lot simpler if everybody was trustworthy.

Last But Not Least

Unfortunately, bash and other shells consider a null path element to be the current directory. For example, this setting: PATH=/bin: (note the trailing colon) is equivalent to PATH=/bin:. Likewise, PATH=:/bin is equivalent to PATH=.:/bin. **Be sure you don't have any stray colons in your PATH setting.**