Homework 3

Due: May 9, 2016 **Points:** 100

Questions

Remember to justify your answers where appropriate.

1. (*10 points*) An attacker breaks into a Web server running on a Windows 10-based system. Because of the ease with which he broke in, he concludes that Windows 10 is an operating system with very poor security features. Is his conclusion reasonable? Why or why not?

2. (*20 points*) This exercise has you practice with *nmap*, which was discussed in section last week.

   (a) Using *nmap*, please scan the host `spc.cs.ucdavis.edu`. Which network servers are accepting connections at ports below 5001? Along with your answer, please turn in your output from *nmap*, show how you got your answer, and state the time and date on which you did the scan.

   (b) Now do the same with `cowhollow.cs.ucdavis.edu`. What problem did you encounter? How did you get around it?

3. (*30 points*) Consider how a system with capabilities as its access control mechanism could deal with Trojan horses.

   (a) In general, do capabilities offer more or less protection against Trojan horses than do access control lists? Justify your answer in light of the theoretical equivalence of ACLs and C-Lists.

   (b) Consider now the inheritance properties of new processes. If the creator controls which capabilities the created process is given initially, how could the creator limit the damage that a Trojan horse could do?

   (c) Can capabilities protect against all Trojan horses? Either show that they can or describe a Trojan horse process that C-Lists cannot protect against.

4. (*40 points*) This problem asks you to write a program using C or C++ that functions in the same way that a basic anti-virus program functions.

   The file "vdetect.str" contains lines of the following format:

   `name``:`*`string`*

   where `name` is a string that does not contain a ':' and `string` is a string. The string may contain non-printing characters, which will be recorded as '\x*nn*', where *nn* is a 2 digit hexadecimal number. This represents the character sequence that indicate the presence of malware.

   If a line begins with the hash mark '#', it is a comment line and is to be ignored.

   Write a program *vdetect* that reads one or more files and compares each file to the strings in *vdetect.str*. If the file contains any string in *vdetect.str*, print the corresponding `name` and the number of the byte in the file where the string begins. The exit status code is to be 0 if no matches occur, 1 if any match occurs, 2 if there is a malformed line in *vdetect.str* and no matches are found, and 3 if there is a malformed line in *vdetect.str* and a match is found.

   The command line syntax is:

   `vdetect [ -q ] [ -s ] [ -d` *`strfile`* `] [` *`file1`* `[ ... ] ]`

   where *`file1`*`, ...` is the list of files to scan (if none, read from the standard input). Options are:

`-d` *`strfile`*	Use *strfile* rather than *vdetect.str*
`-s`	Stop scanning the file when the first match is detected
`-q`	Do not print any output; just give the exit status code

   If your program finds a malformed line in *vdetect.str*, it is to print the line number and a message like this:

```
line 13: Malformed line in vdetect.str
```

If you use another string file (that is, give the `-d` option), put that file name in place of "vdetect.str".

If your program cannot open a file, use *perror*(3) to print an appropriate error message.

Remember to write your program robustly! Check the return values of library functions and system calls, and so forth. Also write your code modularly; we may, for example, ask you to change from string matching to pattern matching in a future assignment.

Extra Credit

5. (*10 points*) Discuss controls that would prevent a bacterium from absorbing all system resources and causing a system crash.