Homework 1

Due: Wednesday, January 22, 1997 at 11:59PM

Analytical

1. One author has stated that the four key concepts of security are preventing interruption, interception, modification, and fabrication. Are these concepts equivalent to the three discussed in class (preserving confidentiality, integrity, and availability)? If so, which ones; if not, is there any relationship between the sets of concepts?
2. The importance of security is often a function of time; as time passes, the need for confidentiality (for example) may diminish. Present a situation in which the confidentiality of data should be preserved for at least 10 years, and explain why. Then present a situation in which the confidentiality of data need not be preserved for more than 48 hours, and explain why.
3. Someone asks you, "Since the HRU result says that the security question is undecidable, why do we waste our time trying to figure out how secure the UNIX operating system is?" Please give an answer justifying the analysis of the security of the UNIX system (or any system, for that matter) in light of the HRU result.

Programming

4. Write the program described in the attached manual page. Submit it with a Makefile and a README describing any compile-time options. Please be sure the output, and the actions, match the manual page description; if something is not specified, however, you may handle it in an manner you consider reasonable (but please point out what you did in the *README*)! The usual rules of good programming style (comments, error checking, and so forth) apply. Your program should run on the DECs and HP's in the CSIF (you may require compile-time options to be set).
   Hint: Watch out for invalid file names and file names or user names that are very long.

Extra Credit

5. Modify the program so that if no mode is specified, the program tests and reports on all the modes.
6. The HRU result shows the set of safe protection systems is not recursive. Show it is recursively enumerable.

NAME

access – check accesses

SYNOPSIS

access *file mode* [*user* [.. *user*]]

DESCRIPTION

access says whether the named *user*s can access the *file* in the given *mode*. *User* must be avalid user name or a UID; access checking is based on the UID in all cases. The following *mode*s are recognized:

–r read

–w write

–x execute

–c create; this is false unless the user can write to, and search, the directory containing the file. If the user can do so, and the file does not exist, it is true; if the user can do so and the file exists, the user must be able to delete the file.

–d delete; this is false unless the user can write to, and search, the directory containing the file and the file exists. The program honors the sticky bit semantics of systems such as HP/UX.

EXAMPLES

Can the users bishop and holly read the file /home/heberlei/notes?

```
example1% access /home/heberlei/notes -r bishop holly
bishop can read /home/heberlei/notes
holly cannot read /home/heberlei/notes
```

Can the users bishop and holly execute the file /pkg/bin/ksh?

```
example1% access /pkg/bin/ksh -x bishop holly
bishop cannot execute /pkg/bin/ksh
holly cannot execute /pkg/bin/ksh
```

Can the users bishop and holly create the file /home/seclab/bin/newprog?

```
example1% access /pkg/bin/ksh -x bishop holly
bishop cannot execute /pkg/bin/ksh
holly cannot execute /pkg/bin/ksh
```

SEE ALSO

chmod(1), *ls*(1)

NOTES

Suppose a user wishes to delete a file *x* from a directory *dir*. If *dir* has the sticky bit (permission bit 01000) set, then the user must own *x* in order to delete it. Of course, the user must also be able to write to *dir* as well.

The SunOS, Solaris, IRIX and Ultrix versions of the UNIX operating system honor these semantics; HP/UX does not.

AUTHOR

your name goes here