

CSIS0230A Principles of Operating Systems (Class A)

Tutorial 12

Unix Security

The following program is written as a simple message board system. It is stored in the file `/root/msgboard.cc`. Your task will be to setup the system and remove security problems in it. The program is shown below. Try to understand it fully before proceeding.

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cstring>
#include <cerrno>
#include <unistd.h>
#include <wait.h>
using namespace std;
const char* DATA_FILE = "/var/local/message/data";
inline void die(char *msg) { cerr << msg << strerror(errno) << endl; exit(1); }

int main() {
    for (;;) {
        char c = 0;
        cout << "Read messages, Write a message or Quit (r/w/q)? ";
        cin >> c;
        while (cin && (c != 'r' && c != 'w' && c != 'q')) {
            cout << c << "Invalid input! Try again (r/w/q): ";
            cin >> c;
        }
        if (!cin) return 0;
        cin.ignore(1000000, '\n'); // eat the remainder of the line
        if (c == 'r') { // use "less" to display message board
            if (fork() == 0) { // child
                execlp("less", "less", DATA_FILE, 0);
                die("Cannot exec: ");
            }
            wait(0);
        } else if (c == 'w') {
            cout << "Type your message. Press enter 3 times when finished\n";
            ofstream of(DATA_FILE, ios::app); // in append mode
            if (!of.is_open())
                die("File open failed: ");
            for (int entercount=0; entercount<3; ++entercount) {
                if (!cin.get(c)) return 0;
                of.put(c);
                if (c!='\n')
                    entercount--;
            }
        } else if (c == 'q')
            return 0;
    }
}
```

The idea is that every user should be able to read the message board, although modifications

should be allowed only through the `msgboard` program, which should only allow users to read messages from and append messages to the message box, nothing else.

1. Set up the system by compiling the program, copy it to an appropriate position (`/usr/local/bin/msgboard`), create an empty file (`/var/local/message/data`) and change the permission modes for them. The data file should be readable by anybody, but writable only by `root`. Make sure that a normal user can run the program correctly. (Hint: you need to set the `setuid-bit` for the compiled `msgboard` program.)

The following steps check whether your system has been setup correctly:

- a. Login as normal user (`os`)
 - b. Try to overwrite the data file by `cat > /var/local/message/data`. (A normal user should not be able to arbitrarily write data file.)
 - c. Run the `msgboard` program and add a new message. (You should be able to add a new message while using a normal user account)
 - d. Use the same program to view the message. (You should find your message have been added to the message board)
2. There is a security hole in the system: the program tries to execute `less` to show the message box to the user, but the program executed might not really be `/usr/bin/less`. A normal user can subsequently execute any program with root privilege. Perform such an attack. After that, try to remove the security hole.
 3. There is another problem in the system: `less` is too powerful a program to be used this way. Read the man page of `less`, in particular the section about security. What type of attacks can be done to the current system? How to solve the problem? (Hint: is special privilege really needed for reading the message board?)

Further discussions (think after the tutorial):

- The current system has a race condition: if two users try to write a message at the same time, the message board can get corrupted. Discuss how to deal with the problem by using file locking `flock(2)`.
- Naively using `flock(2)` may make users wait for unnecessarily long amount of time if they want to write to the file. Explain why, and suggest how temporary files can be used to deal with the problem.
- Suppose you want to use a temporary file in `/tmp`. How you can avoid two users using the same temporary file? (See man page of `mktemp(3)` and `mkstemp(3)`.)