CS50 Trust Models

Overview

Downloading a piece of software from the Internet requires a substantial amount of trust on part of the user. The user must trust that the piece of software that is being downloaded doesn't contain malicious code. In theory, any software downloaded onto a computer could delete all of the files on that computer. Yet, we still trust that the software we download is safe and secure. This is the basis of **trust models**.

Key Terms

- trust model
- backdoor

Backdoors

To the right is an excerpt of a hypothetical login program written in C, which checks a username and password to determine whether a user's account credentials are valid. In reality, login programs would probably compare the user's inputs against username and password values stored in a database. Furthermore, these would most likely be encrypted in some way and not just stored as plain text. Still, we'll use this simplified version for the sake of example.

```
if ((strcmp(username, "rob") == 0 &&
       strcmp(password, "thisiscs50") == 0) ||
    (strcmp(username, "tommy") == 0 &&
       strcmp(password, "i<3javascript") == 0))
{
    printf("Success!! You now have access.\n");
}
else if (strcmp(username, "hacker") == 0 &&
       strcmp(password, "LOLihackedyou") == 0)
{
    printf("Hacked!! You now have access.\n");
}
else
{
    printf("Invalid login.\n");
}
```

Notice that after performing the initial check for username and password combinations, the code offers an additional way to gain access to the system (by using the username "hacker" and the password "LOLihackedyou"). This method of accessing a system through an alternate means, one that differs from the way that users are supposed to access a system, is known as a **backdoor**.

In this case, any users who were to read the code of the login program would be able to identify the fact that there's a backdoor into the system. However, users who download software usually won't have the opportunity to see the code of a program before it's compiled.

Exploits in a Compiler

Even if a user sees a program's code before they download it and determines that there doesn't seem to be any malicious code or backdoors in the code, that doesn't necessarily mean that the program itself is secure. Compilers, the program that translates source code into object code, can also be the source of exploit.

There are a couple ways that compilers can be used to exploit users. A compiler could, for instance, be programmed to take a perfectly benign login program and inject code into it that creates a backdoor. Anyone who looked at the source of the login program code itself wouldn't detect any signs of a backdoor. But if the source code were compiled with the malicious compiler, then the resulting program would have the backdoor in it. Of course, in this case, anyone who were to look at the source code of the compiler would see that there was code in the compiler that injects malicious code into the login program.

Let's take this one step further. Imagine that we wrote a compiler that injected malicious code into the compiler itself (remember that compilers themselves need to be compiled). Then a hacker could theoretically take benign source code for a compiler and turn it into a malicious compiler. In this case, even if the compiler source files and the login program source files didn't contain any malicious code or backdoors, compiling the source files could still result in the injection of malicious code.