**HOMEWORK 13**

**Introduction:** A compiler is written in C.

The code in the C compiler interprets the character escape sequence. It knows in a completely portable way what character code is compiled for a new line in any character set. The act of knowing, then allows it to recompile itself.

Suppose we wish to alter the C compiler to include the sequence "\v" to represent the vertical tab character. We then recompile the C compiler, but we get a diagnostic. Obviously, since the binary version of the compiler does not know about "\v," the source is not legal C. We must "train" the compiler. After it "knows" what "\v" means, then our new change will become legal C. We look up on an ASCII chart that a vertical tab is decimal 11. We alter our source. Now, the old compiler accepts the new source. We install the resulting binary as the new official C compiler and now we can write the portable version.

Again, in the C compiler, the routine "compile" is called to compile the next line of source.

A simple modification to the compiler is done that will deliberately miscompute source whenever a particular pattern is matched. If not deliberate, it would be called a compiler ‘bug’. Since it’s deliberate, it’s called a ’Trojan Horse’.

The bug planted would match code in UNIX ‘login’ command. The replacement code would miscompile the login command so that it would accept either the intended encrypted password or a particular known password. Thus if this code were installed in binary and the binary were used to compile the login command, the attacker could log into that system as any user.

The final step is to simply add a second Trojan horse to the one that already exists. The second pattern is aimed at the C compiler. The replacement code is a self-reproducing program that inserts both Trojan horses into the compiler. This requires a learning phase. First we compile the modified source with the normal C compiler to produce a bugged binary. We install this binary as the official C. We can now remove the bugs from the source of the compiler and the new binary will reinsert the bugs whenever it is compiled. The login command will remain bugged with no trace in source anywhere.

**1) Describe briefly and clearly how the attack works**

The attack allows for injection of code into the compiler and then concealing its tracks. In the case of Ken Thompson, he injected code into the compiler that matched the UNIX login command but would accept the user's encrypted password OR a particular password that he had set up. That backdoor enabled him to log into the system as any user.

**2) If you suspect that your machine has been compromised, what should you do about it?**

If you suspect that your machine has been compromised, you should probably use a new system. As Ken Thompson explained, it is nearly impossible to find his attack, even using normal debugging methods, so it would be a lot easier/faster to just replace your system. Obviously it depends on what was compromised: if the C compiler was compromised, use a C compiler that you can trust, etc.

**3) What other kinds of software like compilers do we usually trust that have the potential to be compromised?**

IDEs that call the compile chain for the user - IDEs could very easily insert code into the compile chain without the user knowing about it.

Device drivers.