The most famous story of a Trojan horse involved Odysseus of Homer’s epic poem, the Odyssey. The Greek siege of Troy had lasted for ten years. The Greeks devised a new ruse: a giant hollow wooden horse. It was built by Epeius and filled with Greek warriors led by Odysseus. The Trojans were convinced by a Greek spy, Sinon, to bring the horse into the defended city. The Trojans hugely celebrated the end of the siege, so that, when the Greeks emerged from the horse, the city was in a drunken stupor. The Greek warriors opened the city gates to allow the rest of the army to enter, and the city was pillaged ruthlessly, all the men were killed, and all the women and children were taken into slavery.

**Def:** A *Trojan horse* is a program with an overt, documented or known, effect and a *covert*, undocumented or unexpected, effect.

The term Trojan horse was first used in 1974 when talking about computer malware. A simple example of a Trojan horse would be a program named “waterfalls.scr” claiming to be a free waterfall screensaver which, when run, instead would allow access to the user’s computer remotely.

Trojan horses can make copies of themselves. One of the earliest Trojan horses was a version of the game *animal*. When the game was played, it would create an extra copy of itself. These copies spread, taking up much room.

**Def:** A *propagating Trojan horse*, also called a *replicating Trojan horse*, is a Trojan horse that creates a copy of itself.

Thompson, et. al. added a Trojan horse to the login program. When a user logged in, the Trojan horse would accept a fixed password as well as the user’s *normal* password. Anyone reading the source code for the *login* program would instantly detect this Trojan horse. So to obscure the code, Thompson had the compiler check the program being compiled. If that program was *login*, the compiler would add the extra code to use the fixed password. Hence, no code needed to be added to the source code for *login*. Thus, an analyst inspecting the source code for *login* would see nothing amiss. If the analyst compiled the *login* program from that source, she would believe the executable to be uncorrupted.

This extra code was visible in the compiler source. To eliminate this problem, Thompson modified the compiler. This second version checked to see if the compiler, actually the C preprocessor, was being recompiled. If so, the code to modify the compiler so as to include both this Trojan horse and the login Trojan horse would be inserted. He then compiled the second version of the compiler and installed the executable. He then replaced the corrupted source code with the original source code of the compiler. As with the *login* program, inspection of the source code would reveal nothing amiss. However, compiling and installing the compiler would insert the two Trojan horses.

Thompson’s point is that no amount of source-level verification or scrutiny will protect you from using untrusted code.