CSE 509 - System Security
Assignment 1 : Reflections on Trusting Trust

ANIRUDH KULKARNI

In this paper Ken Thomson, one of the masterminds behind the Unix operating system, discusses in an elaborate manner the implications of trusting code that you did not create yourself. Ken portrays how he infused a bug into a compiler. He explains this in three stages. In each stage he showcases a concept and uses code samples in C to illustrate the same.

Firstly, the author introduces self-reproducing program - a computer program which takes no input and produces a copy of its own source code as its only output. He concludes by stating that a program can be easily written by another program. In addition, it can also contain an arbitrary amount of excess baggage that can be reproduced. Secondly, the author makes use of the escape characters in C to illustrate a way in which the source code of a compiler can be altered. Thirdly, the author illustrates how login in UNIX can be intercepted by pattern matching to gain access to the system. However, he disregards this stage when it is not supplemented with techniques in stage 1 and stage 2, as it will easily raise suspicion. Finally, he explains how all the above stages can be used in tandem to insert Trojan horses (deliberate misconduct) for exploitation of the system.

In conclusion, he states that you can't confide in code that you didn't absolutely make yourself. The source level verification does not assure you safety. Additionally, these kind of attacks are not restricted to a particular programming language or compiler. These attacks can occur at any level - assembler, loader, or hardware microcode. In some cases, it can become impossible to identify a well planted bug. Also, the author criticizes the attitude of the public (press and the law makers) towards the hackers and explains the gravity of such acts by drawing analogies.

QUESTION 1 : Since we can inspect the source code of the compiler to see whether it injects backdoors in the compiled programs, what is the point of this article?

One can check the source code of the compiler to rectify and delete the buggy code, however the binary of the compiler can be injected with self reproducing segment and installed as official compiler. In such cases, no matter what changes are made to the source code, the new binary will reinsert the bus whenever it is compiled.

QUESTION 2 : Can we be 100% sure that a PHP/Python/NodeJS web application is benign if we just inspect its source code?

No. We cannot be fully sure about this.

- Most of the scripting language interpreters (including the ones states in the question) are written in low level languages. The binary files of these interpreters can itself be bugged with a self producing malware.
- There can be many malicious programs written at lower levels which might be impossible to find.