# Reflections on Trusting Trust- (Kartikeya Bhardwaj id: 1840684)

### *A high-level summary of the paper and a concise recap of its main take-away*

This paper is about the acceptance speech of Ken Thompson's Turing award. Ken Thompson is most known for writing UNIX and he is being given the Turing award for the same. But he decides not to talk about UNIX and take credit for it as it is an effort of a lot of people. Instead he decides to talk about a program. He decides to do it in three stages. In the first stage, he explains the college game of a self-reproducing program, i.e., a program that prints itself. The program he shows is a C program with the body of the program in a string variable. In the second stage, he explains the "chicken and egg" problem related to the C compiler. The C compiler is written in C itself. He shows that to update a new case he first has to "train" the old binary. Once the new source has been compiled, the new binary can be used as new version of the compiler which contains that case. In the third stage, he explains how a security bug can be planted without any source code for it. He does this by first compiling a buggy version of the login command. This buggy version is now made the standard C compiler. Later, when someone finds out this bug, it is fixed but the new binary will reinsert the bugs whenever it is compiled. The message he wants to take home is no code can ever be trusted. No amount of scrutiny will protect you from using untrusted code. He explained that he had just picked on the compiler. If he had gone lower to an assembler, a loader or hardware microcode these bugs would be harder to detect. He goes on to criticize the press that hacking should be criminalized as someone robbing your house.

### *A description of what you most valued learning from the paper and the paper's greatest strengths*

The point that really struck me was no amount of scrutiny will protect you from untrusted code and a security flaw can be present in the hardware microcode too. This stuck with my head because lately I was reading an article about how a tiny chip was used to infiltrate top companies.

I don't know about the authenticity of the news to be honest but this seemed like what Ken Thompson was talking about. A security bug can be present anywhere and there is a great need to ensure for program correctness.

### *A discussion of what you disagreed with or felt could be improved to make the paper stronger*

A point where I felt the paper missed is in stage 3 Ken had said the login command will remain bugged with no trace of it anywhere. But wont the compiler released with the new source have it fixed? And this version of the compiler can be stated as a critical version and sent to consumers. I felt the paper did not elaborate on how to fix this problem clearly.