Diverse Double-Compiling in a CI/CD environment

Ludvig Christensen, Supervisors: Aman Sharma, Deepika Tiwari
Master thesis@CHAINS project

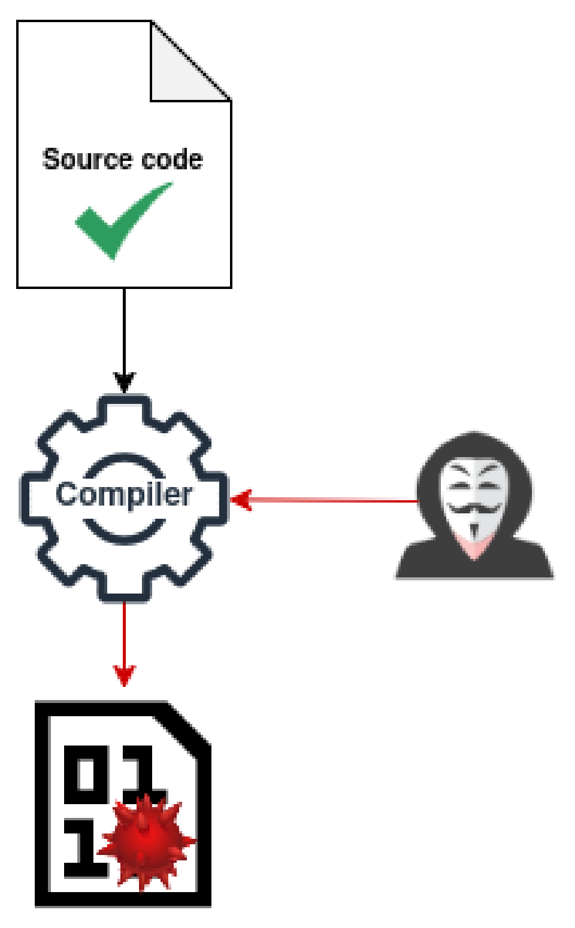
How can you trust software?

"You can't trust code that you did not totally create yourself."

"No amount of source-level verification or scrutiny will protect you from using untrusted code."

- Ken Thompson in *Reflections on Trusting Trust* [1]

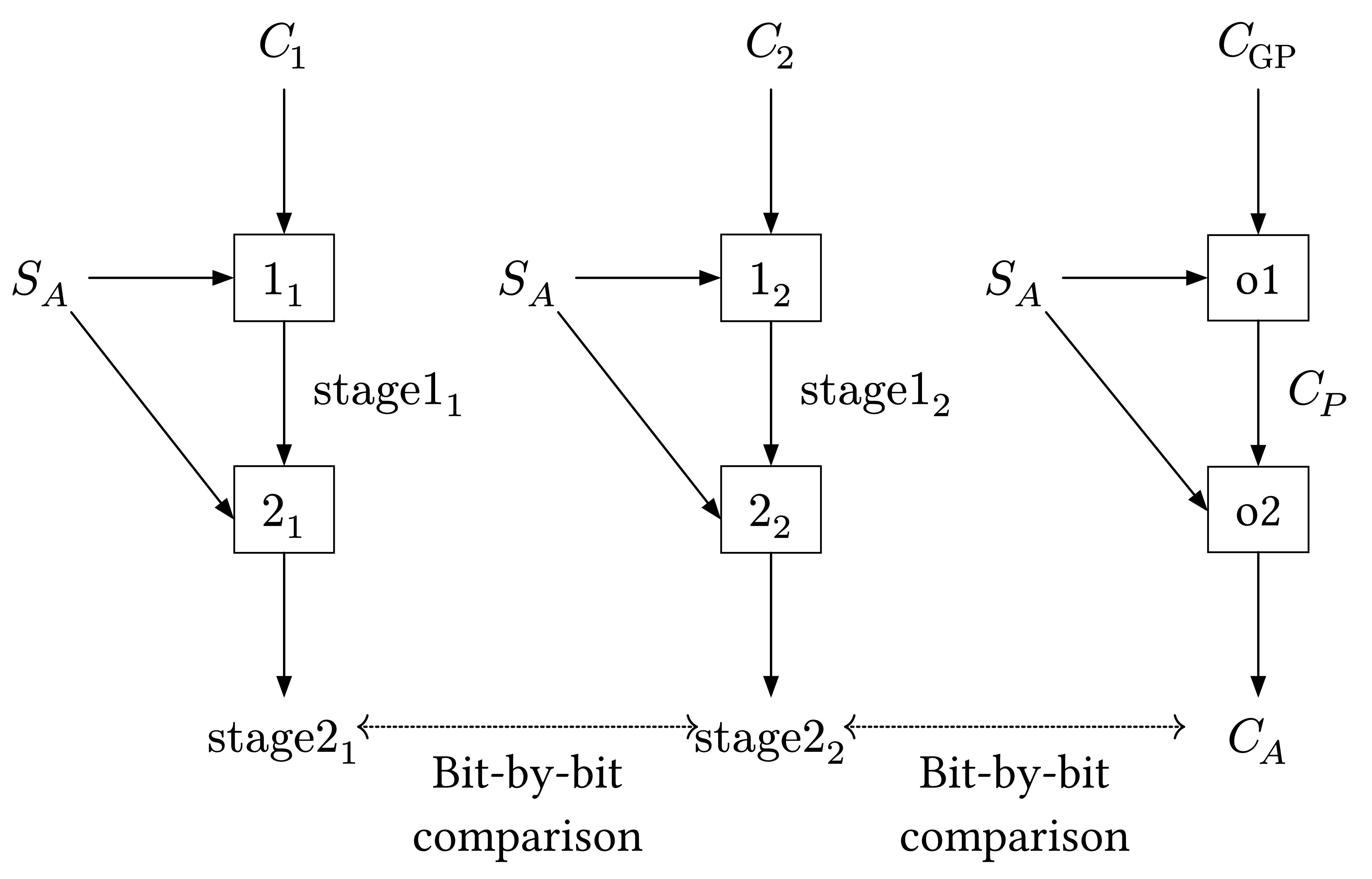
Trusting Trust Attack[2]



1. An attacker modifies the compiler binary to contain a self-replicating trojan.
2. The malicious compiler trojan inserts itself when compiling its own source code and backdoors some other target source, for example the Unix login command.

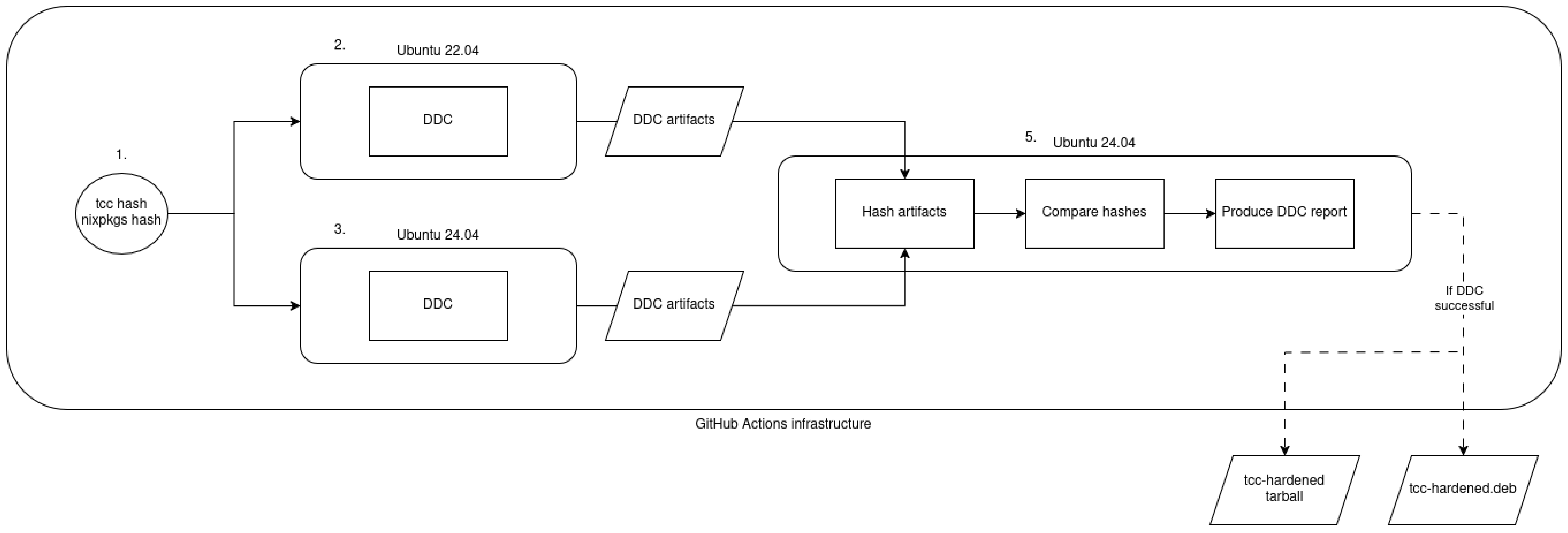
Problem: Since the trojan only resides in the binary, inspecting the source code will not detect it.

Diverse Double-Compiling [3]



- $C_1, C_2, C_{\mathrm{GP}}$: Potentially malicious compilers
- S_A : Source code of C_{GP} or its next version
- Compile the source S_A twice, first with each potentially malicious compiler, then again using the result of the first compilation

ddc4cd - implementation



Motivations for DDC

- Combine the trust of multiple compilers instead of trusting one
- DDC increases trust that a binary corresponds to its alleged source code

Contributions

- Automating the DDC process using modern CI/CD practices
- Release *tcc-hardened*: .deb archive and tarball
- Explore increased diversity in the DDC process

Bibliography

[1] K. Thompson, "Reflections on trusting trust," *Communications of the ACM*, vol. 27, no. 8, pp. 761–763, 1984.

[2] N. Rosencrantz, "Diverse Double-Compiling to Harden Cryptocurrency Software." 2023.

[3] D. A. Wheeler, "Fully Countering Trusting Trust through Diverse Double-Compiling." 2009.

Project repository: https://github.com/chains-project/ddc4cd/