DDC4j

Diverse Double-Compilation 4 Java

The trusting trust attack [1]

How can we tell if a binary contains a backdoor?
Oftentimes we **can't**.

If the compiler is infected, then **any program compiled with it** might be.

javac is written in Java, so it compiles itself.

So javac could reinfect itself every time a new version is compiled!

This is the **trusting trust** attack.

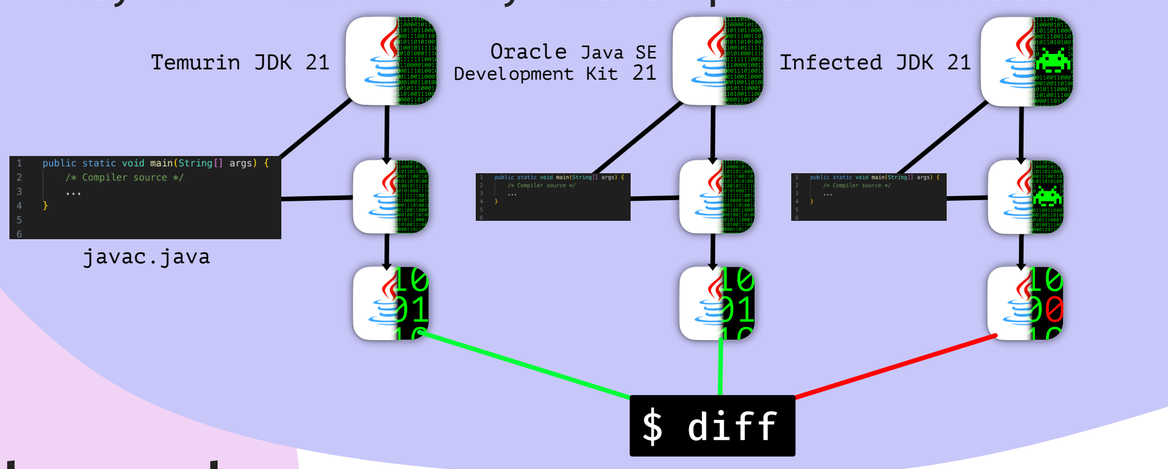
DDC

[2] Double-Compilation

A way to **detect** the trusting trust attack.
It requires javac to be deterministic - given the same input, it must produce the same output.

Luckily, javac is sufficiently deterministic!
(as of JDK 21)

Now, we compile javac using different compilers - they will diff if any one compiler is infected.



Diversity through bootstrapping

But Java only has **one** modern compiler - javac!
What if **all** modern compilers have the same self-replicating virus?

We need a more **diverse** compiler. How can we get diversity?

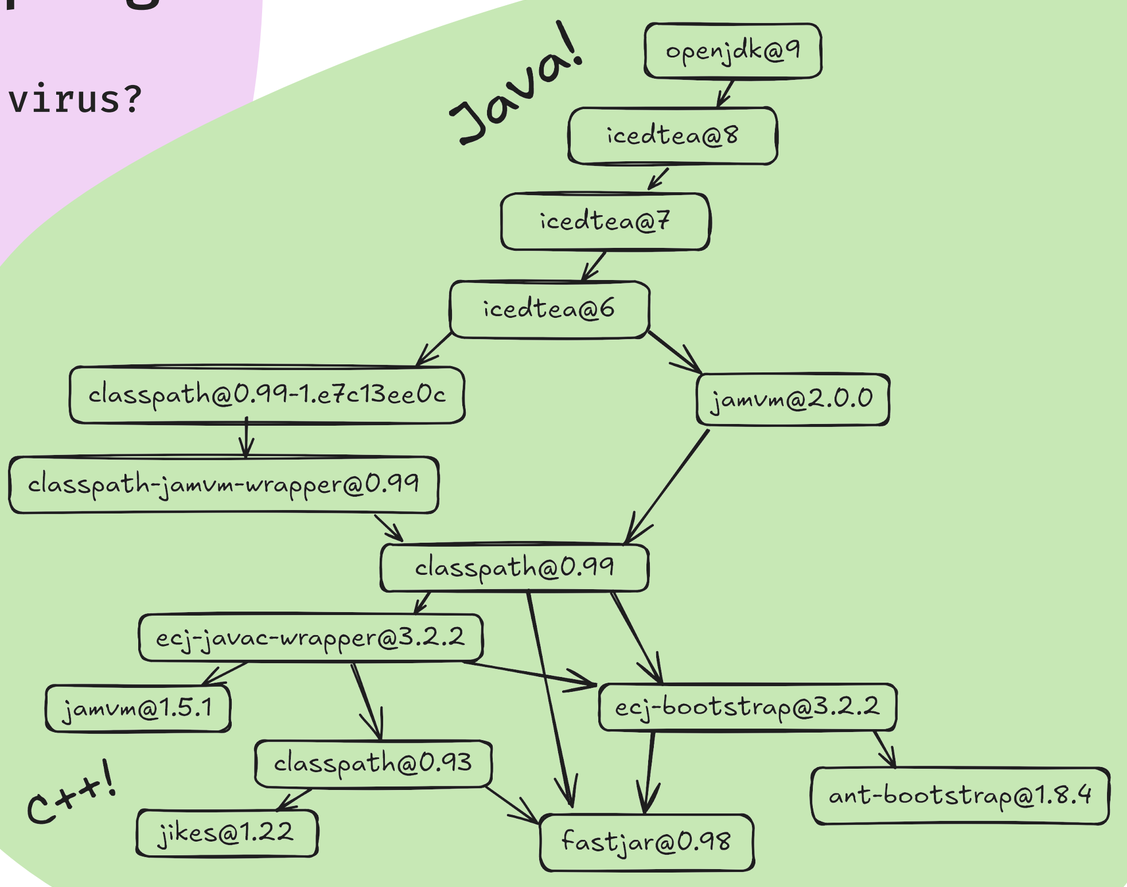
- Using another Java compiler (e.g. ECJ) - doesn't work
- Writing our own Java compiler - too complex

The only feasible option: bootstrapping Java from C++.

Bootstrappable Builds have completed a bootstrap chain before - we can use their compilation chain. [3]

Using their recipe in GNU Guix, we can build javac from C++.

```
$ guix install openjdk@21.0.2 --no-substitutes
```



Sources

[1] Ken Thompson, “Reflections on trusting trust,” Commun. ACM, vol. 27, no. 8, p. 761–763, Aug. 1984.

[2] David A. Wheeler, "Countering trusting trust through diverse double-compiling,” in 21st Annual Computer Security Applications Conference (ACSAC’05), Dec. 2005, pp. 13 pp.–48.

[3] Ricardo Wurmus / Bootstrappable Builds, “From C++ to Java — bootstrappable.org,” Dec. 2018.

Eskil Nyberg
eskilny@kth.se

Elias Lundell
ellundel@kth.se