

Verification Conditions

CS 6V81-002: Language-based
Security

Jan 9, 2008

The Problem

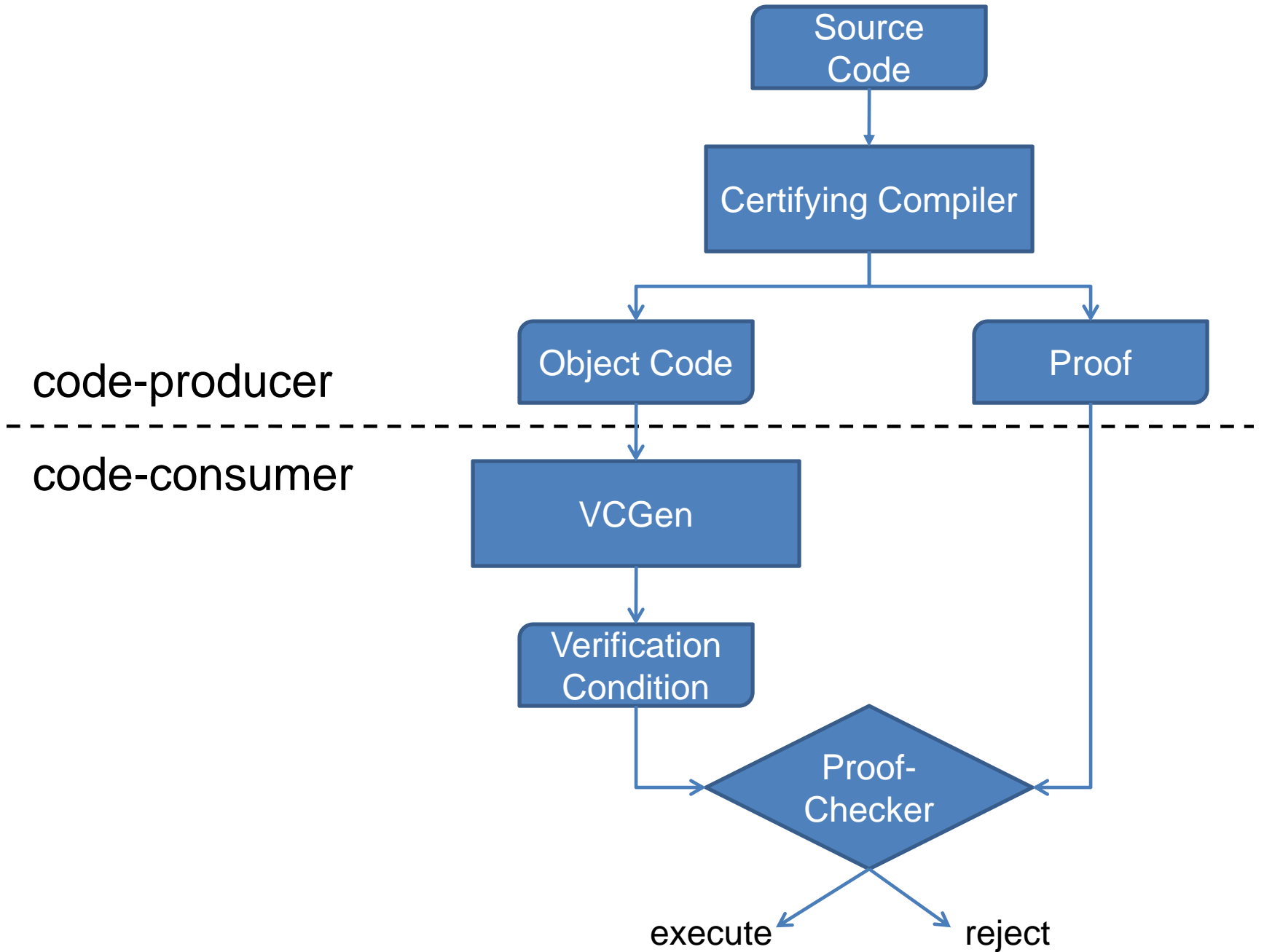
- Two parties: code-producer, code-consumer
- Code-producer
 - has created some policy-adherent code
 - wants to give it to code-consumer
 - is unwilling to divulge source code
- Code-consumer
 - wants to execute producer-supplied code
 - does not trust code-producer
- How can code-producer convince code-consumer that the code satisfies the policy?

Non-solutions

- Digital signatures
 - require consumer to trust producer (or some third party)
 - third-party signing just begs the question: How can code-producer convince third-party?
- Code review
 - prone to error, expensive, time-consuming, ...
- Antivirus Scanning
 - Only enforces trivial policies like, “Code must not be one of the executables in my signature list”
 - Cannot (precisely) enforce policies like memory safety

Proof-Carrying Code

- Code-consumer automatically extracts from the object code a logical predicate called a **verification condition**
 - if vc is true, then code satisfies security policy
 - generating a vc is easy (linear time)
 - proving that the vc is true might be very hard (undecidable in general!)
- Code-producer gives consumer a proof of the verification condition
 - Why might it be easier for producer to generate such a proof?
- Code-consumer verifies proof
 - verifying a proof is easy (linear in proof size)



Possible Attacks

- What happens if...
 - code-producer sends invalid proof?
 - code-producer generates valid proof, then alters object code maliciously?
 - code-producer sends valid proof, but it proves policy-adherence for some other program?
 - a man-in-the-middle tampers with the code or the proof
 - a man-in-the-middle tampers with the code AND the proof

Today: Introduction to Verification Condition Generation