A Versatile Code Execution Isolation Framework with Security First

Johannes Krude Ulrike Meyer

Research Group IT Security
RWTH Aachen University

Motivation

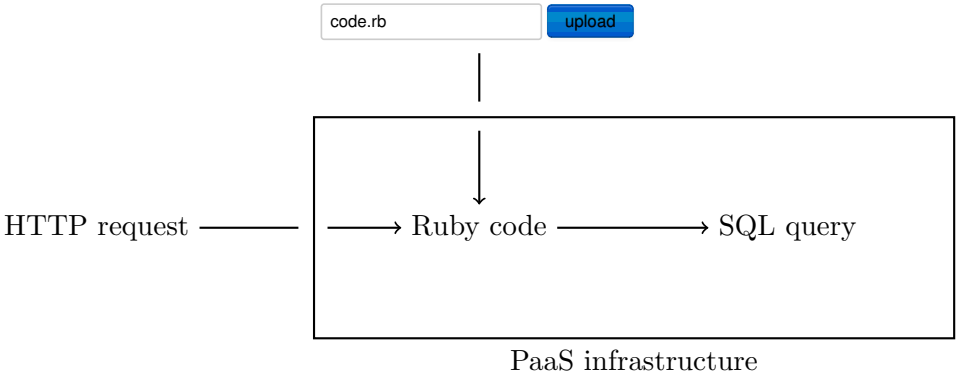
isolating untrusted code execution

should be:

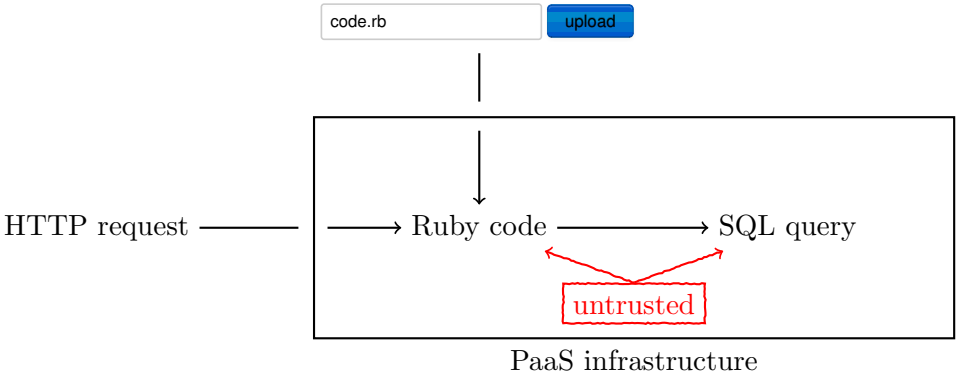
dead simple

should allow for:

Example 1


code.rb
upload
HTTP request
Ruby code
SQL query
PaaS infrastructure

Example 1


code.rb
upload
HTTP request
Ruby code
SQL query
untrusted
PaaS infrastructure

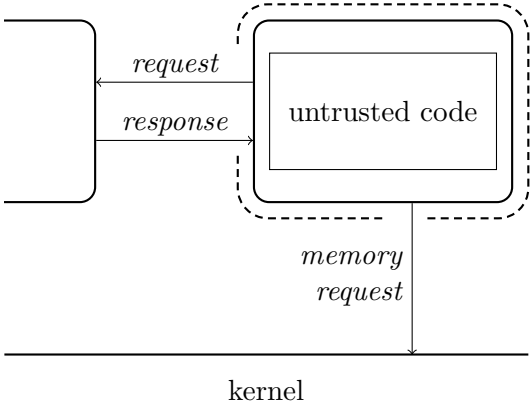
Dead Simple

memory protection:

process barrier

system call access:

seccomp filter based syscall policy



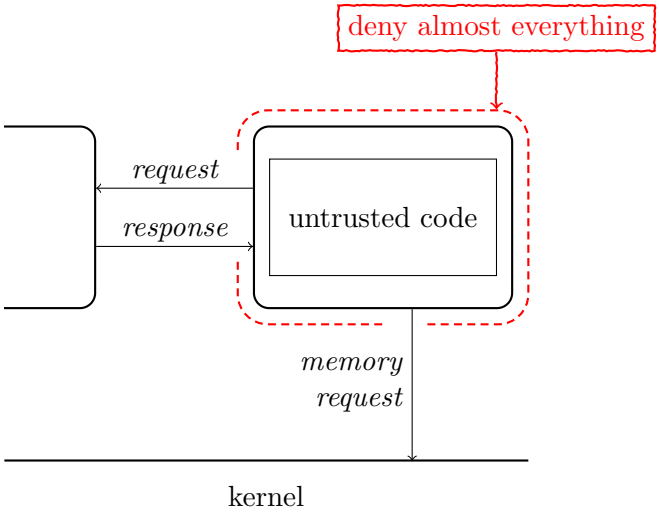
Dead Simple

memory protection:

process barrier

system call access:

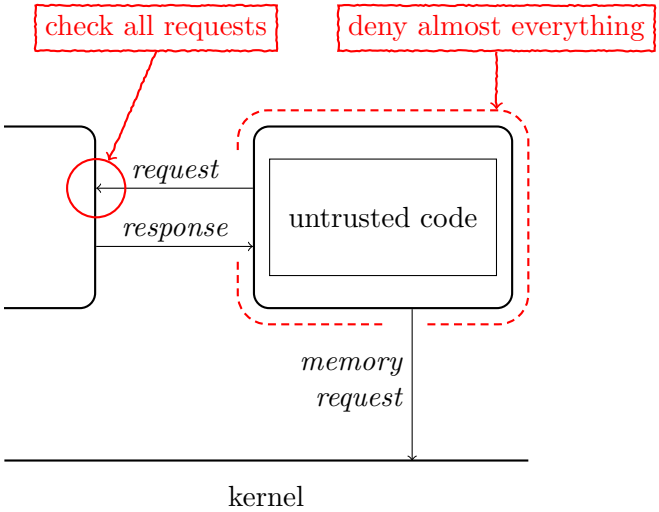
seccomp filter based syscall policy



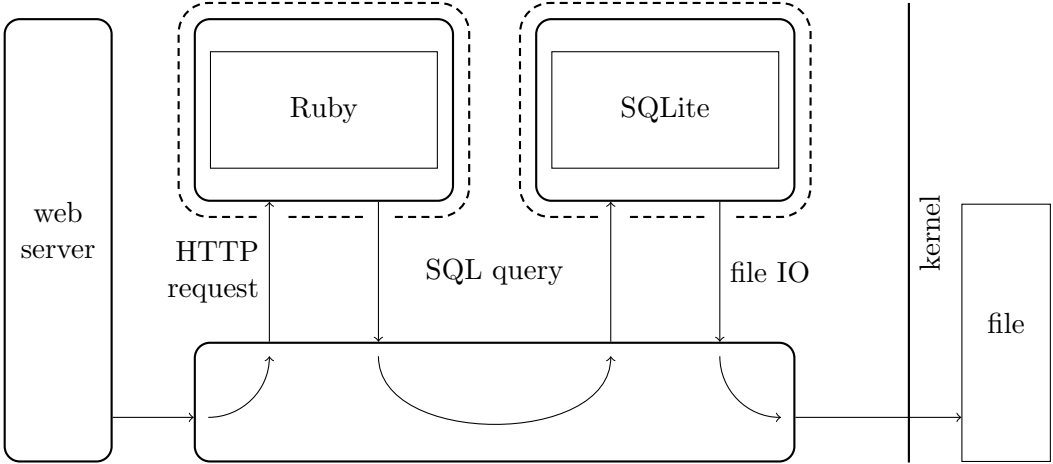
Dead Simple

memory protection:
process barrier

system call access:
seccomp filter based
syscall policy



Example 2


web server
Ruby
SQLite
HTTP request
SQL query
file IO
kernel
file

Evaluation

plain computations: no slowdown

Ruby in isolation: $< 100KiB$ memory overhead

isolation setup time: $< 10^{-8}s$

Key Ideas

simple OS features

process barrier

system call policy

control all access path

memory protection

request delegation

try it out: `http://sandbox.itsec.rwth-aachen.de/`

Questions?

System Call Policy

disallow everything except:

- `write` on output pipe
- `read` on input pipe
- `poll`
- `brk`
- anonymous `mmap`
- `mremap`
- `munmap`

Isolation Setup Sequence

1. (S) `pipe`
2. (S) `clone`
3. (I) `close`
4. (I) `execve`
5. (I) `setrlimit(RLIMIT_AS)`
6. (I) `prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER)`

Request Delegation