Web-basierte Systeme – Übung

X2: WebAssembly outside the browser

Wintersemester 2023

Arne Vogel
Security & Portability

- Implemented with security in mind!
- Use WebAssembly outside the browser?
- Protective barriers to system’s resources
- Kernel mediates system resource access
- Each system has own system calls
Abstraction

- Interface to system resources
- At compile time different implementations are used
- Problem solved?
Abstraction

- Interface to system resources
- At compile time different implementations are used
- WebAssembly bytecode is platform independent
- Where does the interface implementation come from?
In the Browser

- Emscripten emulates POSIX on the web
- JS glue code uses browser APIs
- Browser talks to kernel
Portability

- Compile for every system you target
- More work for every systems
- Do we know all systems?
- Systems that do not exist yet?
Portability

- Only one compilation target
- Compile against the WebAssembly System Interface (WASI)
  - Functions are left to be imported
- Runtime provides system specific implementations
  - Imports are provided at instantiation
- Program has access to all system calls
- No limitation on arguments
- Limit system calls
  - e.g., only read and write files
- Limit arguments
  - e.g., only read files in directory `/config/wbs`
  - e.g., only write files in directory `/tmp`
- Runtime itself can limit capabilities to functions
Usecases

- Two broad categories
- Standalone applications
  - e.g., compile *grep* for universal deployment
- Libraries
  - Execute WebAssembly inside of programs through embedded runtime