CS161 Fall 2026 — Introduction to Computer Security — Exam Prep 9

Q1 *Isolation: Boxed In* 📦 **(13 points)**

EvanBot is designing how to run untrusted code, and considers three layers of isolation: the **operating system** (isolating processes), the **hypervisor** (isolating virtual machines), and **WebAssembly** (isolating modules within a single process).

Q1.1 (3 points) An operating system protects three kinds of state using three different isolation techniques. Match each kind of state to the technique the OS uses to protect it.

Registers in the CPU:	◯ Time multiplexing	◯ Translation	◯ Emulation
Main memory:	◯ Time multiplexing	◯ Translation	◯ Emulation
I/O devices:	◯ Time multiplexing	◯ Translation	◯ Emulation

Q1.2 (3 points) Under virtual memory, the OS translates a virtual address by looking up its page number in the process's page table and copying the offset (the bottom 12 bits) unchanged. Firefox's page table contains:

Virtual Page Number	Physical Page Number
`0x00004`	`0x60C25E6`
`0x00005`	`0x1F96EC7`
`0x00006`	`0xEC70DB7`
`0x00007`	`0x69977244`

Firefox reads from virtual address `0x00006A3C`. What physical address does the OS access?

This content is protected and may not be shared, uploaded, or distributed.

Q1.3 (3 points) WebAssembly runs many untrusted **modules** inside a single OS process. Each module is given its own contiguous 2^{32}-byte chunk of the process's 2^{64}-byte address space. Which of the following mechanisms help prevent one malicious module from reaching into another module? Select all that apply.

- ☐ Each module can only form 32-bit addresses, so it cannot name any address outside its own 2^{32}-byte chunk.
- ☐ Wasm has no arbitrary jump instruction; programs may only jump to a small set of pre-defined targets.
- ☐ Compiled module code is immutable and lives outside the 2^{32}-byte data chunk, so a module cannot rewrite its own code.
- ☐ Virtual memory (VMEM) gives each module its own page table, isolating their address spaces.
- ◯ None of the above

Q1.4 (1 point) Is WebAssembly's module isolation **stronger** than the OS's process isolation?

- ◯ Yes, because each module gets its own page table, which the OS does not give to individual processes.
- ◯ Yes, because a module cannot make arbitrary jumps, so no memory-safety exploit is possible inside a module.
- ◯ No, because Wasm gives up some isolation strength in exchange for much cheaper isolation within a single process.
- ◯ No, because each Wasm module is run in its own OS process, so its isolation is exactly the OS's and no stronger.

Q1.5 (1 point) Naive software interposition can be defeated by an attacker who `jmp`s **past** an inserted bounds check. Does Wasm prevent this bypass?

- ◯ Yes, because Wasm has no arbitrary jump instruction — a program may only jump to a small set of pre-defined targets, so there is no way to land just after a check.
- ◯ Yes, because a module's compiled code is immutable, and immutable code cannot be jumped over.
- ◯ No, because a module may still jump anywhere it likes, as long as the target lies inside its own 2^{32}-byte chunk.
- ◯ No, because Wasm does not insert checks at all — it isolates modules by emulating every instruction instead.

Q1.6 (2 points) Emulation and software interposition both add isolation checks to untrusted code. State the key difference in **when** each performs its checks, and name the resulting drawback of each. (In 10 words or fewer.)

This content is protected and may not be shared, uploaded, or distributed.

(Question 1 continued…)

This content is protected and may not be shared, uploaded, or distributed.