CS161 Fall 2026 — Introduction to Computer Security — Discussion 10

Q1 *Isolation Potpourri* **(3 points)**

Q1.1 (1 point) True or false: In virtual memory, two processes can use the same virtual address and refer to different physical memory locations.

Q1.2 (1 point) True or false: Software interposition checks while running the code while emulation rewrites code before it runs.

Q1.3 (1 point) Which security principle is best represented by OpenSSH's isolation design of using the Monitor Box, Authentication Box, and User Box?

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

Q2 *Plug It In* **(4 points)**

PlugIn is a company whose products allow customers to upload third-party plugins (compiled to WebAssembly) that run inside a single long-lived server process. For example, each plugin handles a different user's data, Plugin A handles User A (Alice), Plugin B handles User B (Bob), and so on.

Assume the system has complete OS-level isolation between the PlugIn server process and everything else.

Q2.1 (1 point) Plugin A and Plugin B both run inside the same OS process and share the same virtual address space.How does WASM prevent Plugin A from reading Plugin B's memory?

Q2.2 (1 point) Mallory, a malicious user, noticed that Bob (another user) has a bank account access function that lives at a known offset in their module's memory block region. Mallory attempts to write Wasm bytecode that computes that address and jump directly into it mid-execution. Why does this attack fail?

Q2.3 (1 point) Mallory found that the previous attack didn't work, so instead she attemps to write bytecode with only valid jumps within her own module, passing compilation successfully. Next, during execution, Mallory attempts the following:

- Mallory first writes a carefully crafted sequence of raw bytes into her own memory region that happen to encode a raw x86 jmp instruction pointing into Bob's bank account access function
- Mallory then attempts to overwrite her own compiled x86 function call with a pointer to those bytes, so that the next time that function is called, it will jump into Bob's module.

Why does this attack fail?

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

Q2.4 (1 point) Assume the only 2 users are Mallory and Bob, both have access to the function `get_time()` which can read the server's clock, and each call allocates 512 MB of memory. Mallory is still a valid user, how can Mallory exploit this feature?

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