HOMEWORK 2

Anirudh Pal (pal5)

Answer 1: Each process should have its own runtime kernel stack separate from its user stack. We can understand the importance of this with the help of an example.

1. Imagine there are two processes A and B. Let's say that A and B are malware and also share memory space.
2. If A where to make a sleep() call it would go from current to sleeping in Kernel mode.
3. Then B while A is sleeping can modify A's stack so the return from sleep points to malicious code.
4. Now when A wakes up, it will lead to sleep() returning to malicious code that runs in Kernel mode and thus breaks Isolation/Protection.

This similar to what we did in Lab 1 malware section and thus is a critical feature of modern kernels.

Answer 2: Virtualization is the concept of running multiple operating systems on a single CPU or physical system. Full virtualization requires that there is a single OS in a multi-OS system which is at a higher privilege level than the guest OSes. Such an OS is often called the hypervisor. The hypervisor often has to virtualize several hardware features that might be different for different guest OS like EFLAGS.

Best Case Overhead: In the best case, the guest OS runs app code with no overhead from the hypervisor as it is also reactive.

Worst Case Overhead: In the worst case, the guest OS runs a system call which traps to the hypervisor, then returns to the guest OS kernel portion and after running some non-privileged instructions, it runs a privileged instruction which again traps to the hypervisor. At that point the hypervisor might execute it, virtually execute it or a number of other overhead heavy operations before it returns to the guest OS.

In x86, popf is used to change the flag registers like IF (Interrupt Enable/Disable). When run in Kernel mode it will change IF as requested. When run in User mode it will not change IF as requested and still behave as the instruction was successful. This is problematic in a system with guest OS, as the guest OS runs in User mode. Even though it might want the

Interrupts disabled it will not be able to do so. The hypervisor can detect this easily and thus has to resort to heavy overhead tricks like ‘just in-time compiling’ etc.