Operating Systems – Quiz 3

Name: Aaron Paterson

You wake up with the buzzing of your alarm clock (okay…I know you use your cell phone…) Today is your SECOND interview with Apple Mac OS Division and with Craig Federighi. Apple is looking to enhance the way they handle security for their November 10th conference next week.

To answer this exam, you need to answer the question as you would in a job interview. Write as if you were answering a question that someone verbally asked. That means you need to explain your stance and provide evidence that your stance is the proper stance. Diagrams and drawings can help clarify if they are used properly.

1.) Craig asks you: "First let's understand your philosophy on operating systems…My question is…what is the role of an Operating System in regards to security?"

An operating system needs to be able to differentiate between users and programs and limit their privileges accordingly. For example, if I save my private information on my computer, the operating system has to both make sure it is reliably available to me in the future, but also to make sure it is not available to anyone else. To accomplish this, it is equally important for an operating system to have good program design, as it is to have an easy to use and effective updater. The updater is also an example of a program the operating system uses to install other programs, which has the security obligation of verifying that the user actually wants to install that program, and limiting its privileges accordingly. This is something existing operating systems are notoriously bad at; trojans and ransomware are only possible because of faults in the implementations of this design.

Another example of a place an operating system can fall short on security is the clipboard. The clipboard seems like a helpful feature, it made life easier for the first person that came up with it for sure. But an unforeseen consequence of giving every computer user on earth the ability to dump data into easily accessible memory is that passwords, credit card numbers, and other can easily be stolen, possibly without even exploiting any code bugs. So to be secure an operating system needs to be able to effectively test new features without immediately immortalizing them into users' workflows, which is something else they have fallen short on in the past.

2.) Craig says "That's interesting…now let's talk about your actions…What single change would you make to any Operating System to make it more secure? What you mention may already exist…that's okay, I just want to see what YOU would do if you had a magic wand and WHY you would do it…"

I think sandboxing is the natural conclusion of operating system security, as having a sandboxed running environment has numerous other advantages in addition to the added security. Sandboxing is currently a focal point of enterprise computing, there are several third-party sandboxing solutions that have entered the operating system scene very aggressively. (Docker, flatpak, and snap to name a few.) Sandboxing allows a user to run vulnerable code without putting their private data at risk, which I think should be the assumption by default, as vulnerabilities are almost never known about by a user before they are exploited.

It also seems like it could greatly enhance dependency and package management, as sandboxing allows developers to be sure that the environment that they test their program on will be identical to the one their program runs on. This improves compatibility by eliminating compatibility updates, because multiple containerized versions of previously incompatible program environments can coexist on the same system. Really once you accept the idea of sandboxing, it seems like a design fault for an operating system to run code from different developers in the same environment at all, because all that can do is cause problems, without providing any real advantages.

Plus the concept is already so ubiquitous in programming and math; a lexical scope in a programming language, or a universe in mathematics, are examples of conventions that restrict access to information to an application specific set, to great effect. So, the current mode of installing and running programs seems more appropriate for debugging an OS than for production to me, and currently package managers and library linkers really just paint over this flaw.