it contains a bug that permits either of these. For example, QEMU uses dynamic binary translation to achieve acceptable speed, and if this process were compromised or a fault were exploited, the software could become unstable or breach security safeguards.

If users will accept the speed penalty of emulation when security can be absolutely guaranteed, then why not build an emulator on the most widely deployed and secure virtual machine, the Java VM (JVM)? The JVM has been tested for over 10 years as a secure means of running code, and users are often content to let unvetted code downloaded from the Internet execute within the Applet Sandbox, the security container supplied by the JVM. This is because the JVM guards against fundamental programmatic errors, such as accessing arrays in memory beyond their valid size and reading data from unallocated memory. Further, a JVM with a security manager installed to enforce a sandbox can veto any sensitive operation attempted by guest software.

JPC is just this: an x86 PC emulator written entirely in Java. There is no native code in JPC; it emulates all the standard hardware components of an x86 PC while remaining entirely inside the Java Applet Sandbox. Thus within these security restrictions the x86 operating systems and software running inside JPC are totally isolated from the underlying hardware, even to the point that said hardware does not need to be an x86 PC.† From the host's point of view, JPC represents just another Java application/applet, and thus the host can be confident that the code (whatever it may be) is safe to run. JPC can boot DOS and modern Linux, giving the guest software and OS complete unfettered access to all the hardware of the virtual machine, including root/admin access, all while staying inside the Java sandbox.

To break out of JPC, an attacker would have to find a bug in JPC's code coinciding with a bug in the JVM, which could enable a sensitive action on the host computer that was also within the powers of the user running the JVM. This represents a breach of three completely independent layers of security. Each layer is typically built by completely different companies, and because each layer is needed in so many different circumstances, their security is under constant independent testing and review. Note also the coincidence requirement in breaking the layers. It is not sufficient to find a bug in JPC and then move on to the task of breaking out of the JVM; the hacker needs to find a bug in JPC that directly (and already) connects to a suitable security bug within the JVM being used. As JPC is open source, it is a relatively simple task to review the code and build a "clean" version, and in high security applications a security-hardened JVM could be used. JPC thus presents an impregnable barrier to malicious x86 code, and indeed provides the most secure, convenient, and safe way to examine malicious x86 code in action.

Like virtualization, as hardware speed increases, the applications of emulation can expand from the security area to mission-critical systems that need to ensure robustness. No matter how the emulated machine crashes (or how hard), the host machine is unaffected and can continue

† JPC has been adapted to run inside a J2ME environment and can then boot original and unmodified DOS on an ARM11-based mobile phone!