

Extended Abstract:
“Novel Applications of Xen: Virtual Training and Malware Evaluation”
Stephen Brueckner
ATC-NY

Xen is commonly used by enterprises to virtualize server environments. For the past three years we have used the Xen hypervisor for two unusual applications: a virtual training environment and a testing platform for malware evaluation. Both of our projects employ user-configurable heterogeneous networks of customized VMs, and users need to simultaneously access both hypervisor controls and guest internals. We describe our applications, their requirements, and the tools we developed to meet those requirements. Our purpose is to provide feedback to and foster discussion with the Xen developer community.

Our Cyber Defense Trainer (CYDEST) provides automated, web-accessible training to students in both computer network defense and live forensics. It features a number of virtual networks spanning multiple physical machines which are abstracted to a single network for the student. We have built a custom VM management interface that focuses on the student’s situational awareness rather than the traditional system administrator’s point of view. Our interface graphically displays network topology, including internal VM information such as hostnames, IP addresses, and OSs. The interface also includes novel concepts of VM status that are user-oriented, e.g. a VM is not considered “up” until it is accessible via ssh and VNC. Finally, to support our training model, the interface is browser-based so students may conduct their training online and without special client requirements.

Another challenge for CYDEST has been to automate the monitoring and control of VMs’ internal states. To avoid interfering with the student’s training scenario, our monitoring and control traffic uses an out-of-band network that employs a separate physical and virtual network. To minimize our footprint inside the VMs and to maintain functionality in the absence of a network connection, we access guests using the virtual serial console. We have built an application that wraps the “xm console” command to support interaction with CYDEST’s parallel, automated processes. Challenges include queuing multiple simultaneous access requests, signaling timeouts of processes both inside and outside the VMs, buffering of I/O for non-human console users, XML encapsulation of output (e.g., stdout vs. stderr), and accommodating various login prompts and user names.

Our second project is the Exploit and Malware Incubator (EXAMIN), a testing and reverse engineering platform for isolating, triggering, detecting and evaluating stealthy malicious code such as rootkits. We note that a common anti-tamper technique used by malware authors is to deactivate when inside a virtualized environment. However, this technique is unlikely to remain tenable in the future, as VMs are becoming increasingly common as deployed environments. Since we focus on detection of stealthy malware, which often require unadulterated kernels to properly execute, we employ Xen’s HVM capability for EXAMIN rather than running paravirtualized VMs.

EXAMIN's incubator is a heterogeneous VM network that is customizable by the user to resemble deployment conditions as closely as possible. To support this, we have created a Virtual Network Builder (VNB) tool that greatly reduces the amount of time required to deploy VMs, as well as the level of expertise needed to do so. It includes a GUI front end to edit the virtual network's topology and allows deep internal customization of VMs, including network settings, user account configuration, and the installation of software packages. The back end accordingly modifies "vanilla" images of Linux and Windows VMs into custom images, using a combination of dead image manipulation and live VM configuration.

A significant component of EXAMIN is the use of VM introspection to provide high-assurance security monitoring of guests. VM introspection is the process of accessing a guest's state from the hypervisor, and therefore does not rely on the integrity of the guest's OS to provide accurate results. We have implemented a number of introspection-based services using integrity checking and cross-view monitoring techniques. One significant challenge of VM introspection is bridging the "semantic gap," i.e. translating the low-level view of a guest's virtual hardware into high-level OS semantics that can be intelligently reasoned about. We have developed a technique for bridging the semantic gap that is automated and generalized (for both Linux and Windows), such that code written to run inside a guest can be executed in dom0. This technique eliminates the need for developers to learn a new language or API to conduct VM introspection, and greatly eases the porting of existing applications from guests into dom0.

Our hope in describing our projects' usage of Xen is to foster discussion on a number of topics. We present our concerns with Xen's isolation guarantees and the consequences of Xen's rapid development, such as changing APIs, emerging tools, and lack of documentation. We have a "wish list" of features such as faster guest access channels and multiple serial consoles. Finally, we request input on our tools and approaches, inviting ideas on alternative techniques and tools.