

## Introduction

Over the past several years there has been a shift in the strategies used by attackers to compromise computers. The focus has changed from penetrating one computer at a time to compromising computers on a mass scale in order to create a mass of zombie computers that can all be used at once to perform large scale attacks. The security applications that work to defend most computers typically serve in a passive mode, providing passive monitoring and reacting when a scan results in a virus, malware, etc. There are obvious flaws in this type of security defense and the shift in security technology is to develop and use more active monitoring to prevent attacks from taking place by adding hooks to the system being monitored. *Lares: An Architecture for Secure Active Monitoring Using Virtualization* offers a solution to achieve active monitoring by utilizing the benefits of virtualization.

## The Goal

So why virtualization? A fundamental problem with having a security tool that runs on a client machine is that most worthwhile attacks will set out to disable any security systems present on the machine. Many security mechanisms attempt to prevent themselves from being disabled, such as an anti-virus actively working to ensure that the anti-virus process is not terminated. But what happens when the attacker has complete control and administrative access on a client machine? The anti-virus can hardly prevent itself from being uninstalled by an attacker with administrative privileges. One must assume first and foremost that all areas of a client computer have been completely compromised and then work towards designing a security system that can still thwart an attacker. This is where virtualization can provide an advantage over traditional technologies. Utilizing the advantage that a virtual host has to look inside a virtual machine from the perspective of a higher domain, the virtual host proves to be an ideal location for introspection and offers unique security advantages. The virtual machine itself is not even aware that it is running on a virtual host. The virtual host has access to the virtual machine but the virtual machine does not have access to the virtual host (Figure 1).

The goal of *Lares* is to provide an active monitoring solution that meets the following 4 requirements:

*“The first requirement states that an attacker should not succeed in circumventing hooks or generating spurious notifications. The second requirement ensures that an attacker cannot modify the context information  $I_e$  before invocation of  $N_e$  to alter or hide information regarding the event  $e$ . The third requirement ensures that the functionality of the security tool itself is not maliciously altered, defeating all attacks that tamper with the application process or any underlying subsystems it depends on. The fourth requirement ensures that the responses on the state of the system are always carried out as intended without letting the attacker modify them.”*

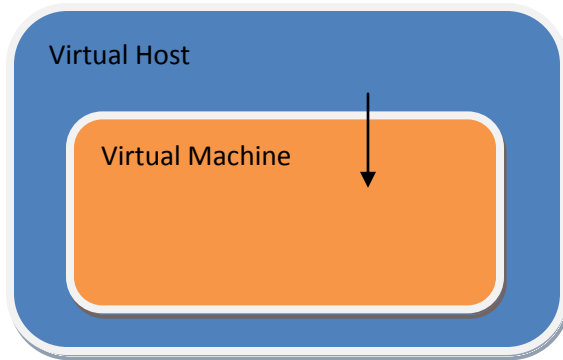


Figure 1

## The Technology

*Lares* is built on top of the Xen hypervisor (freely available at [xen.org](http://xen.org)) and uses Fedora 7 (freely available from [fedoraproject.org](http://fedoraproject.org)) as the security virtual machine. The client machine used for design and testing within *Lares* is a Windows XP Pro SP2 virtual machine. The security vm is part of the trusted computing base (TCB) of the hypervisor and the guest vm has no access to the security vm. Hooks can be placed at any point within the client machine so that when a part of the guest vm is executed that contains a hook the systems control flow diverts to the trampoline. The trampoline acts as the go between for the guest vm and the security vm (see Figure 2 from *Lares: An Architecture for Secure Active Monitoring Using Virtualization below*).

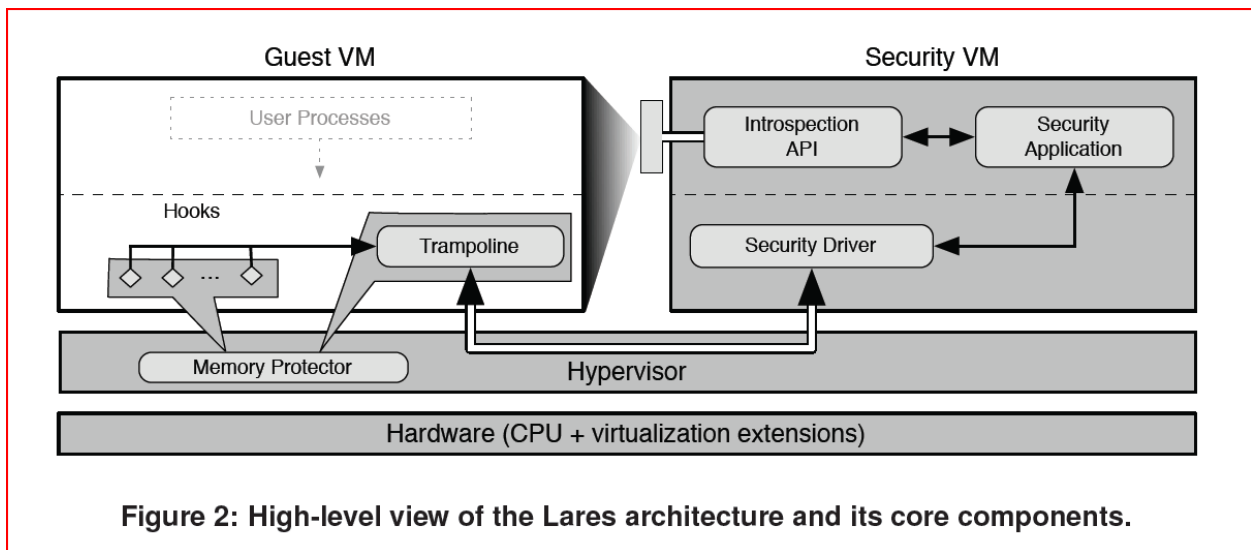


Figure 2: High-level view of the Lares architecture and its core components.

The advantage offered by virtualization is that specific parts of the guest VM's memory can be declared read-only, ensuring the integrity of the hooks and trampoline. Since this write protection occurs outside the guest VM at the hypervisor level there is no way for an attacker to disable or modify the hooks or trampoline from within the guest VM. When a hook is triggered, information about the process is sent by the trampoline through the hypervisor to the security driver on the security VM. Here the security application on the security VM can analyze the information sent by the trampoline and determine what actions, if any, need to be taken. Since the security VM is part of the TCB it has the ability to analyze aspects of the Guest VM such as memory and the Guest VM hard drive since they are located on top of the hypervisor. This is another advantage of using virtualization over traditional technologies.

## The Analysis

As with any security defense a decision must be made to decide if the performance trade off is worth the added security or not. Lares does offer a performance trade-off because execution on the Guest VM is halted while the Security VM analyzes the data sent by the trampoline. A greater performance hit occurs when introspection of the Guest VM is used on top of just notification via hooks. This does introduce a level of concern that the security application must run quickly and efficiently in order to keep the performance hit from being noticeable. Another concern is that *“the overall performance of a given application will ultimately depend on the number of hooks it uses in addition to its use of introspection and other techniques used to collect data for processing each hook.”* Careful care must be taken by the security application developer to not use unnecessary hooks and spend minimal time processing the information provided by the trampoline. However, these performance costs in a large environment are most likely less than the performance cost of running standalone security mechanisms on each individual Guest VM.

One example of added security is that Lares provides protection from rootkits. The fundamental goal of a rootkit is to disable or alter the hooks put in place by security mechanisms. Since Lares is able to protect the memory of the Guest VM where the hooks are located, rootkits are unable to alter or disable the hooks. This is a significant advantage over traditional security mechanisms.

Extra care must also be taken to keep the Security VM itself from being compromised since it is part of the TCB. This should be easy enough to accomplish since the Security VM itself does not have a compelling reason to be on a network and can be isolated from other computers with relative ease, but it is worth mentioning since a compromised Security VM would be a significant point of failure. Imagine a Security VM that simply does not respond to calls from the trampoline. This could result in many frozen Guest VMs waiting for the Security VM to respond.

A current concern among engineers of virtual environments is the amount of resources wasted by installing an anti-virus application on every client on a virtual host. The overhead of running the anti-virus application is multiplied by running the same application on every VM in the environment. This is a major concern in large virtual environments. Lares illustrates an example in which the processing of an anti-virus application can be done on a central Security VM and the overhead of running a full anti-virus application can be avoided on each Guest VM. Lares also illustrates how the hypervisor can be used to isolate and protect specific parts of a Guest VM by identifying critical parts of the Guest VMs memory as read only.

Another component of the design of Lares worth mentioning is that the initial design imposes a limit on each VM can only be assigned a single CPU core to prevent against a few attacks that could be used if a guest had access to multiple cores. This could cause issues in large scale deployments of enterprise systems that require multiple CPU cores to function efficiently, such as a database server.

Finally, it would be interesting to test and see if a Guest VM could flood the Security VM by repeatedly calling a process with hooks associated with it. Would it be possible to execute an attack that would disable or significantly degrade the performance of the Security VM?

## **Conclusion**

Virtualization opens the door for new security mechanisms that go beyond the capability of traditional mechanisms. As the adoption of virtualization increases daily, it is only natural that new security mechanisms that incorporate and utilize the benefits of virtualization will be developed. Lares illustrates some of the benefits that virtualization offers and provides for a more efficient solution than using stand alone security mechanisms on each client in a virtual infrastructure.