Short Paper: Policy Driven Virtual Machine Monitor for Protected Grids

Fabrizio Baiardi, Laura Ricci
Dipartimento di Informatica
Università di Pisa
largo Pontecorvo, 3 - 56127 Pisa
{baiardi, ricci}@di.unipi.it

Paolo Mori, Anna Vaccarelli
Istituto di Informatica e Telematica
Consiglio Nazionale delle Ricerche
via Moruzzi, 1 - 56124 Pisa
{paolo.mori, anna.vaccarelli}@iit.cnr.it

Abstract

This paper advocates virtualization technology as a methodology to solve the security problems that an organization has to face when contributes with its resources to a grid. In particular, this technology makes it possible to increase the overall security of any system by inserting a set of controls into the code that implements one virtual machine. In this way, a secure cooperation among virtual machine can be implemented. This generalizes the current approach that exploits virtualization only for the confinement of alternative programming environments resulting from the partitioning of a physical machine into a set of non cooperating virtual machines. The ability to support cooperation among virtual machines may be used to define networks of cooperating virtual machines to execute distributed applications. The paper describes a general purpose approach to security based upon virtual networks of cooperating virtual machines and applies it to one of the most challenging problems: that of securing a grid environment.

1. Introduction

Grid environments support the dynamic sharing of computing resources among a large set of entities such as companies, universities, research centers, and so on [1], [2], [5]. These entities belong to distinct administrative domains, and no a priori trust relationships may exist among them. A security support for a grid environment has to face a large number of problems ranging from the middleware of a grid node that attacks the application executed on behalf of a grid user, to malware hidden in the application received from a grid user that attacks resources belonging to the node but outside the grid. We believe that the most appropriate strategy to solve these problems is the one that exploits virtualization, a well known technology to implement abstract resources on top of physical ones that is currently exploited to partition a powerful physical machine into a set of virtual machines (VMs) each supporting a distinct user environment. The sharing does not reduce the overall security, because virtulization confines faults within one VM. Furthermore, it is very simple to detect an attempt of a VM to interact with another one. To fully exploit its advantages with respect to the secure sharing of a set of resources, virtualization should be generalized to virtual networks, i.e. to networks of interconnected and cooperating VMs hosted by distinct physical nodes of a computer network. In this way, several distributed applications can be executed in a secure and confined way on top of a set of network nodes by creating and configuring a set of VMs that are then interconnected into distinct virtual networks, one for each application. Obviously, this poses the problem of a proper compromise between the performance losses due to virtualization with respect to the security advantages in terms of confidentiality, integrity and availability of information and resources. Since we are interested in virtualization as a general-purpose methodology to security issues, we are interested not only in confinement but also in the sharing of resources among VMs of a virtual network. This is important anytime an application is defined by composing modules and service with different security levels, as always happens in grid applications.

Virtualization is a well known technology that has been exploited for a long time. Currently, several tools based upon virtualization are available, among them Xen [3], VMWare [12] and Denali [14]. An attempt to exploit the VM architecture in grid computing is described in [4] and [8] but it is focused on the definition of an architecture that simplify the exploiting of the grid environment by the final user. Instead, this paper is focused on the security aspects of virtual networks of VMs.

This paper is organized as follows. Section 2 describes the VM concepts and how to embed a security support in a VM, Sect. 3 introduces the security requirements of a Grid environment and Sect. 4 shows a first application of these concepts by defining an extension of Xen to support virtual networks of Xen VMs.

1-4244-0307-3/06/$20.00 ©2006 IEEE.

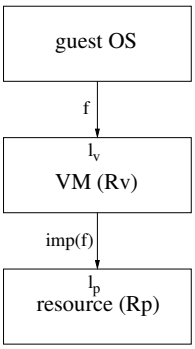

Figure 1. Virtual Machines

2. Security Enhanced VM and VM Monitor

A VM is a set of components to map a program interface Iv, i.e. a programming language, into another interface Ip, to support the execution of programs written using Iv. To this purpose, the VM translates each invocation to a function f of Iv into a sequence $imp(f)$ of invocations to functions of Ip. In the most general case, and the most complex one from a performance prospective, all the translation occurs at run time. Since any interface Ia may be described as a set of resources, Ra, and a set of operations, Oa, on resources in Ra, a VM implements $< Rv, Ov >$ in terms of $< Rp, Op >$. The constrains on Ip to simplify the implementation of Iv have been presented and discussed in [6]. In the case of interest, Iv defines the machine language of a processor, while Ip is the machine language either of the same processor, i.e. Ip and Iv are equal, or of a distinct one. This is shown in figure 1. In the following, the resources in Rv and in Rp will be denoted as, respectively, virtual and physical resources.

From our point of view, the most interesting feature of virtualization technology is that $imp(f)$ may include consistency and security checks on the operations on the resources in Rp. Hence, Iv may be equal to Ip because the VM may have been introduced not only to define a distinct language, but even a more secure implementation of the same one. A case where Iv is equal to Ip arises if we are interested in the secure sharing of a set of resources Rp among a set of user. Here, each user has exclusive access to a distinct VM and Iv is equal to Ip, but each sequence $imp(f)$ should access just a subset of the resources in Rp. Hence, to guarantee the confinement among the VMs, the sequence should includes some checks to prevent a VM from accessing physical resources outside the ones assigned to the user of the corresponding VM. In this way, a VM user can behave as when accessing a dedicated machine, even if other VMs are sharing the same physical resources.

Virtualization requires that the guest OS, i.e. the OS that runs on top of the VM, is transformed to run in non-privileged mode, i.e. user mode. Alternative strategies have been devised to implement this transformation. For instance, some operating system modules may be rewritten so that the VM intercepts their calls, or the VM can dynamically replace the privileged instructions of the guest OS with invocation to its own functions. The latter case exploits an exception handling mechanism where the execution of an instruction of Iv raises an exception handled by the VM by invoking the proper sequence of instructions of Ip. Obviously, the complexity of a VM depends on the architecture of the underlying architecture because some processors provide a powerful support for virtualization, while others, like the IA32 one, offer little or no support at all and a complex software layer has to be developed. However, a powerful support is offered by an increasing number of processors.

From the architecture point of view, to enhance the security of the physical resource, a security component could be integrated in each VM to control the operations that the guest OS invokes on the resource. For instance, a VM can prevent an untrusted application running on the guest OS from transmitting confidential information by controlling the resulting information flow through sockets. In general, distinct VMs can implement alternative security policies and distinct controls for the same policies according to the trust in a user and/or in the application. It should be stressed that some checks to discover attacks against an OS can be executed only through the adoption of virtualization. Consider, as an example, a check on the memory used by the guest OS to discover the signature of some virus or worms that has successfully attacked the OS itself. The check can be simply implemented at the VM level, because the VM can check whether the code of the guest OS is corrupted, while it is rather complex to be implemented at the guest OS level, where the first problem to be solved is the integrity of the OS code that implement the checks.

To increase the overall robustness, a further security layer may be introduced in the architecture. This layer is implemented by a virtual machine monitor, VMMon, that mediates all the accesses of the VMs to the resources, so that a VM can access a resource only if the VMMon has checked the requested operation. In this way, the VMMon multiplexes the physical resources while achieving the logical isolation among distinct VMs. Furthermore, to increase availability and prevent dos attacks, the VMMon should schedule the various requests to guarantee a fair allocation of resources among the VMs. We are interested in the solution where the VMMon runs directly on the hardware, i.e. it is the only component that accesses physical resources.

Besides the VMMon, a supervised mechanism to supports interactions among the VMs on a set of shared resources can be useful even if the VMs do not cooperate to support the same application. In particular, to support grids,

we are interested in file sharing. Consider a computational service with a very large read-only database. The users of this service execute their applications on this resource to exploit this database but, to complete isolate the applications of distinct users, the database should be replicated in each VM, with a great waste of resources. A more efficient solution shares the database among distinct VMs, but this sharing cannot be managed by the VMMon, because it cannot offer the functionalities of an OS as it runs directly on the hardware, Instead, these controls may be delegated to a further VM, FSVM, that runs in user space and physically owns the shared file systems. The security policy of the database as implemented by FSVM specifies the files each VM can access and the operations it can execute on them.

Summarizing, the proposed approach exploits virtualization to define four levels of controls, implemented by: *i)* the guest OS on its users; *ii)* the VM on operations issued by the guest OSs; *iii)* the VMMon on the VMs; *iv)* the VM that owns a shared resource R on the VMs sharing R.

3. Security and Grid

Due to its collaborative and distributed nature, a grid environment requires a complex and powerful support to guarantee the security of its transactions. As a matter of fact, any grid participant shares some resources with any other one, even if no direct trust relationships exist between them. Security management is complex due to the large number of secure relationships among a large number of participants across distinct administrative domains, each with its distinct resources, security architecture and local security policy.

The security model defined by OGSA standards for a grid environment, [9], describes the main security issues and the mechanisms that should be adopted to address them. However, it is well known that current implementations of grid environments do not offer all the mechanisms required to guarantee a complete protection. As an example, Globus [5], currently the widely used grid toolkit, provides a coarse grained control of applications executed on the grid computational resources on behalf of remote grid users. In fact, an application is executed only if the grid user has been authenticated and mapped on a proper local user, but no further controls are executed on the actions performed by the application beside the permission enforcement provided by the OS mechanisms. Some solutions have been proposed to enhanced Globus security, e.g. see [10], [11], and [13].

To show that the virtualization methodology can increase the security of a grid environment, we present a strategy to secure the execution of distributed applications where a job request that submit a distributed application is handled by creating a virtual network. A virtual network is a network of VMs where each process of the application runs in a distinct VM and distinct VMs communicate through the interconnecting virtual network. The VM and VMMon can cooperate to enforce the adopted security policy because any operation attempted by an applicative process on a local resource is controlled by the VM executing the process, while the VMMon controls the interactions among the processes implemented by distinct VMs. This strategy can be easily implemented provided that proper tools to implement virtual networks are available

4. Network-Xen

Network-Xen extends Xen [3] to define a secure virtual network, where VMs control the interactions of processes they run, according to an user defined policy. In the original Xen architecture, Xen supports the execution of several VMs on the same physical machine providing isolation among them. An user may dynamically instantiate a new VM with a guest OS to execute its application. The task of building the initial guest OS structure for a new VM is delegated to a particular VM, Domain0. Proper Xen services run in the Domain0 of each physical node implement the communication network among VMs in the same node. The OSs supported by Xen have been properly modified to run on top of the VMMon instead than on top of the physical machine.

Network-Xen, instead, supports user defined virtual networks to execute distributed applications. To define these networks, it creates the VMs of the virtual network by properly exploiting the Xen services. While the communications among the VMs running on the same node are supported by Xen, the communications among VMs running on distinct nodes are supported by Network-Xen. While, in the original Xen architecture, VMs running in distinct physical nodes communicates provided that the VMs and the Domain0s of each physical machine are in the same network or if the IP addresses are public, in Network-Xen two VMs communicate provided that they are connected in the virtual network that is created. Moreover, Network-Xen provides also a security support that monitors the interactions among VMs.

To show how Network-Xen can be exploited, consider a physical architecture consisting of a cluster of physical machines connected by a local network, and assume that the grid service provider wants to execute several distributed applications simultaneously and in secure way on this architecture. Furthermore, assume that applications should be isolated from each other and that they can share resources such as the file system. When adopting Network-Xen, each physical machine of the cluster runs an instance of Network-Xen, and these instances cooperate through proper services run by their Domain0s. One physical machine of the cluster only, denoted as machine0, is connected to the external network and, consequently, to the grid. Machine0

runs a VM hosting a server that provides an interface to the grid framework. As an example, this server could be Globus, where one of the user defined services creates the network of VMs. The server receives from the remote grid users the job requests, each specifying a distributed application in terms of the processes that compose the application and the communication topology among them. For each request, the server create a proper virtual network by interacting with other Network-Xen servers on the Domain0s of other physical machines. Some features of the virtual network, such as the OS and the memory size of each VM are defined by the request. For instance, the main memory size of a VM depends upon the amount of memory for the corresponding application process. The VMs of the virtual network are mapped onto Xen instances on the cluster physical machines. According to this procedure, a physical machine can host zero, one or several VMs of the same virtual network. A virtualization technology simplifies the dynamic update of this mapping due to performance or reliability reasons, because Xen, as most virtualization tools, supports the migration of a VM to a distinct physical node. While Xen manages communications between VMs on the same physical node through a proper routing service running in Domain0, those between VMs on distinct physical machines are managed by a Network-Xen routing services running in the Domain0s too. These services instantiate a Virtual Private Network among the Domain0s of each couple of physical machines and communications between two VMs are forwarded from a Domain0 of a physical machine to one of another machine only if the two VMs are connected in the virtual network.

In this architecture, one physical node runs the VM that implements FSVM, the shared file system. This machine can be freely chosen, and its IP address must be known by each VM that provides the access to one of the shared file systems. Since the FSVM address does not belong to the subnet of each virtual network, the previously described routing services are exploited also to route the requests for the FSVM address to the proper VM.

The security support of the Network-Xen architecture is implemented by three components. The first component is integrated in each VM, and monitors the interactions between the guest OS and the physical resource. This component has been described in [7]. A second security monitor is integrated in the Xen and Network-Xen routing services to check that the interactions between the VMs do not violate the virtual network topology defined in the job request. In practice, the controls on the network topology are the same ones that a firewall executes when it routes some communication to the hosts it protects. They are currently implemented through the IPQueue library. The third security component is integrated in the FSVM and guarantees that accesses of a grid application to files of the shared file system satisfy the security policy defined by the cluster owner.

References

[1] M. Baker, R. Buyya, and D. Laforenza. Grids and grid technologies for wide-area distributed computing. *International Journal of Software: Practice and Experience*, 32(15):1437–1466, 2002.

[2] S. J. Chapin, D. Katramatos, J. Karpovich, and A. Grimshaw. Resource management in Legion. *Future Generation Comp. Systems*, 15(5–6):583–594, 1999.

[3] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer. Xen and the art of virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2003.

[4] R. Figueiredo, P. Dinda, and J. Fortes. A case for grid computing on virtual machines. In *In Proceedings of Int. Conf. on Distributed Computing Systems (ICDCS)*, 2003.

[5] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In *Proceedings of IFIP International Conference on Network and Parallel Computing*, pages 2–13. Springer-Verlag, LNCS 3779, 2005.

[6] R. Goldberg. Survey of virtual machine research. *IEEE Computer*, pages 34–45, 1974.

[7] F. Martinelli, P. Mori, and A. Vaccarelli. Towards continuous usage control on grid computational services. In *Proceedings of Int. Conf. on Autonomic and Autonomous Systems and Int. Conf. on Networking and Services 2005, IEEE Computer Society*, page 82, 2005.

[8] A. M. Matsunaga, M. O. Tsugawa, M. Zhao, L. Zhu, V. Sanjeepan, S. Adabala, R. J. O. Figueiredo, H.Lam, and J. A. B. Fortes. On the use of virtualization and service technologies to enable grid-computing. In *Proceedings of the 11th Int. Euro-Par Conference, LNCS 3648*, pages 1–12, 2005.

[9] N. Nagaratnam, P. Janson, J. Dayka, F. Siebenlist, V. Welch, S. Tuecke, and I. Foster. Security architecture for open grid service. Global Grid Forum Recommendation Draft, 2004.

[10] L. Pearlman, C. Kesselman, V. Welch, I. Foster, and S. Tuecke. The community authorization service: Status and future. *Proceedings of Computing in High Energy and Nuclear Physics*, 2003.

[11] A. J. Stell, R. O. Sinnott, and J. P. Watt. Comparison of advanced authorisation infrastructures for grid computing. In *Proc. of High Performance Computing System and Applications 2005, HPCS*, pages 195–201, 2005.

[12] J. Sugerman, G. Venkitachalam, and B.-H. Lim. Virtualizing I/O devices on vmware workstation's hosted virtual machine monitor. In *Proceedings of the 2001 USENIX Annual Technical Conference, Boston, Massachusett*, 2001.

[13] M. Thompson, A. Essiari, K. Keahey, V. Welch, S. Lang, and B. Liu. Fine-grained authorization for job and resource management using akenti and the globus toolkit. In *Proceedings of Computing in High Energy and Nuclear Physics*, 2003.

[14] A. Whitaker, M. Shaw, and S. Gribble. Denali: Lightweight virtual machines for distributed and networked applications. In *Proceedings of the USENIX Annual Technical Conference, Monterey, CA*, 2002.