Secure Service-Oriented Grid Computing with Public Virtual Worker Nodes

Matthias Schmidt, Niels Fallenbeck, Matthew Smith, Bernd Freisleben
Department of Mathematics and Computer Science, University of Marburg
Hans-Meerwein-Str. 3, D-35032 Marburg, Germany
{schmidtm, fallenbe, matthew, freisleb}@informatik.uni-marburg.de

Abstract— **Cloud computing uses virtualization technologies to offer a non-shared use rental of computer resources with publicly accessible worker nodes. However, unlike Grid computing, Cloud computing as implemented by Amazon, IBM, Google and Microsoft only offers compute and storage resources from a single organization. Many of the cross-site and cross-organizational advantages offered by Grid computing are lost. In this paper, we present a novel infrastructure that combines the benefits of Grid and Cloud computing: Cheap multi-organizational resources and private compute nodes with root access reachable from the Internet. Our previously introduced virtualization of Grid resources is extended by an approach to offer the same freedom of network access Cloud computing offers, but in a multi-organizational and shared use environment without endangering existing users or resources. An approach is presented for the dynamic network isolation of Grid users from each other as well as a mechanism for shielding the Grid infrastructure from malicious users and attacks from the Internet. This solution overcomes the traditional limitation that Grid worker nodes are kept in private networks and enables new multi-site service-oriented applications to be deployed securely.**

I. Introduction

At present, multi-organizational Grid computing is mainly used by academic institutions with large computational or storage demands. Commercial adoption of multi-organizational Grid computing has been slow, mainly due to security concerns created by the shared use of resources in an environment with unknown users. Furthermore, the fact that Grid worker nodes are usually not reachable from the Internet and thus cannot host service-oriented applications hinders the adoption in the business sector that is heavily interested in the service-oriented computing paradigm. Due to these problems, businesses are turning to the newly emerging Cloud computing paradigm to enact cost saving measures through out-sourcing. Cloud computing uses virtualization technology to offer a non-shared use of computer resources with publicly accessible worker nodes on demand. However, unlike Grid computing, Cloud computing as implemented by Amazon, IBM, Google and Microsoft only offers compute and storage resources from their own (i.e. a single) organization. Many of the cross-site and cross-organizational advantages offered by Grid computing are lost, and years of Grid research are not utilized. In particular, the Cloud computing paradigm does not utilize resource scheduling but offers users near instant access to resources. This forces Cloud computing providers to offer enough hardware to handle unknown peak loads, making Cloud computing more resource hungry and expensive than Grid computing for all applications that do not necessarily need instant access to resources. Thus, while Cloud computing is a great benefit for applications that require instant access to resources, it is a large step backwards for all applications that are less constrained in their execution time (i.e. it does not matter if the application is executed now or in a couple of hours). While current Grid solutions provide such resource scheduling mechanisms, the non-accessible worker nodes and the lack of strong user isolation still hinders the adoption of Grid computing for commercial service-oriented applications.

Simply making the worker nodes of the Grid clusters public is not a viable option, since public worker nodes would clash with the requirements of the traditional batch-job oriented Grid and cluster use. The reason why the batch-job and service-oriented computing paradigms clash is that submitting batch jobs only requires a publicly accessible headnode, while the cluster worker nodes can be operated in a private network, reducing the risk of an external attack, whereas service-oriented Grid applications require a more complex and dynamic setup with accessible cluster nodes, which would also endanger all other users on those resources.

In this paper, we present a novel infrastructure that combines the benefits of Grid and Cloud computing: Cheap multi-organizational resources and private, root-access compute nodes reachable by the Internet. The basic idea of our approach is to virtualize Grid resources to offer the same freedom of network access Cloud computing offers but in a multi-organizational and shared use environment without endangering existing users or resources. This allows service-oriented applications direct, multi-site access to the cluster nodes, without endangering the resources or other computations running on the cluster nodes. To secure the communication between the cluster nodes and protect them from external attacks, the proposed approach provides multi-site, dynamic firewalling together with virtualized compute environments for Grid applications. All Grid services are executed in a personalized virtual environment, namely a Xen virtual machine (VM). All VMs act as cluster nodes and can have public IP addresses. The traffic between the VMs and the access to them is protected by user based, dynamically assigned firewalls. To leverage existing scheduling mechanisms, our proposal is based on the Xen Grid Engine v2 (XGE) introduced in a previous paper [22]. The XGE is an extension of the Sun Grid Engine cluster scheduler [24] and provides the dynamic creation of virtual execution environments for compute jobs.

The paper is organized as follows. Section II states the problem addressed in this paper in more detail. In Section III, related work is discussed. Section IV provides the proposed design. Implementation issues are discussed in Section V. Experimental results are presented in section VI. Section VII concludes the paper and outlines areas for future research.

II. Problem Statement

This paper deals with security issues that we encountered during our work on the German national Grid project D-Grid [1]. The aim of the first phase of the D-Grid project was to link the existing high performance computing centers of German universities and research institutions in a free academic Grid. In the second currently running phase the goal is to extend the Grid to incorporate medium and large scale businesses. Thus, the relationship between Grid users and resource providers changed from highly trusted to mostly untrusted [20]. This change occurred because Grid computing itself changed from a research topic with few users to a widely deployed product with commercial adoption. The traditional open research communities have very low security requirements, while in contrast, business customers often operate on sensitive data representing intellectual property, and thus their security demands are very high. In traditional Grid computing, most users share the same resources concurrently. Consequently, information on other users and their jobs can usually be gained quite easily. This includes, for example, that a user can see the running processes of another user. For business users, this is unacceptable since even the meta-data of their jobs is classified [21]. As a consequence, most commercial customers are not convinced that their intellectual property in the form of software and data is protected in the Grid.

To make the Grid more acceptable to business needs, we previously introduced a solution that makes use of Xen hypervisor based operating system virtualization technology, similar to the approach now used in Cloud computing systems like Amazon's EC2 (http://aws.amazon.com/ec2). This solution lets different users operate concurrently on the same hardware, while isolating them using separate VMs.

Using VMs to enhance security for batch job applications already fulfills the requirements posed by the industrial partners in the D-Grid. All computation takes place inside a shielded environment in which there is no inter-user communication and no connection to the Internet. However, for service-oriented Grid applications consisting of several Grid services potentially from different users executed on multiple sites, using virtual worker nodes without Internet access or inter-node communication capabilities is not sufficient.

A. A Sample Application

The following section presents a sample application that demonstrates the communication requirements of fine-grained service-oriented applications. The application is a multimedia analysis application that runs face and text detection algorithms on confidential video material. The application consists of several Grid services that preprocess and analyze large video files. An input video is split into several smaller parts to facilitate parallel execution of the analysis processes. All worker nodes require a complete installation of the Globus Toolkit 4 (the Grid middleware used in our scenario) in which the computational services are deployed.

The analysis consists of a face detection algorithm that includes the invocation of several other algorithms. Every frame of a video snippet is analyzed to find shapes looking like faces. All occurrences of faces and the lengths of their appearance are stored so that it is possible, for example, to determine the total time different characters appear in the material. Depending on the result of a frame's analysis, some deeper analysis might be needed. For instance, if a face was detected, a face recognition service can be called.

Several VMs are created to handle these tasks, in which the media researcher can install new services autonomously. The number of created VMs is then passed to a Video Splitter service that also runs in one of the created VMs. It then splits the video into as many parts as VMs are available. A face detection service is then run on all VMs. Depending on the results of the face detection run, a deeper analysis is performed. Finally, the partial results of all VMs are collected and merged using a Result Merger Service. For more information on the video analysis algorithms, the reader is referred to [7].

While the application data is secured by using VM techniques, the network and thus the connected resources are fully accessible if no firewalling technology is used. The classical approach to solve this problem is all-or-nothing that permits or allows access to a resource to all users. An Internet connection (if any) is provided by an externally reachable headnode. The same applies to node access. In most cases, users are not allowed to directly log into any of the cluster nodes. This widely used scenario hinders users to utilize the full service-oriented Grid potential. A multi-site, parallel application where all running instances need to share data would not run due to the network restrictions. This could be restricted access to the Internet or only access to a small number of ports on few remote computers. Furthermore, the submitter has no fine-grained control over his or her application. The user could login to the Grid headnode or check the status remotely via MDS but (s)he cannot access a particular cluster node to adjust some settings of a running application instance.

The presented state-of-the-art mechanisms are sufficient for cluster computing scenarios, but inadequate for performing the presented application in the Grid.

B. Threats to Counter

The authors make the standard assumptions seen in most other virtualization security architectures [12], [13]. The hypervisor is part of the trusted computing base (TSB). As we focus on network infrastructure security, we do not deal with attacks against the Virtual Machine Monitor. Building a secure and trusted network infrastructure requires an evaluation of possible attack scenarios. Several threats must be countered:

- **Internal attacks against other users' VMs.** A malicious user could try to remotely compromise other VMs to gain sensitive data or corrupt the work of other users. This can be achieved through the exploitation of software vulnerabilities. To cope with this threat, users must be carefully shielded from each other. By default, no user should be allowed to connect or reach VMs operated by another user. However, user-specified exceptions need to be supported. For example, users from the same virtual organization could request open access between their nodes.
- **Internal attacks against the Grid/Cluster infrastructure.** This includes attacks to corrupt parts of the infrastructure or attacks against particular machines (e.g. the Globus headnode). It must be kept in mind that since users have root access to their virtual worker nodes, they can install privileged software such as DHCP or DNS servers legally. This could enable malicious users to subvert the normal operation of the Grid site and enable inter-user attacks. The security solution must be able to ensure that such infrastructure services do not propagate beyond the users' virtual images and network partitions.
- **External attacks against the virtual worker nodes or infrastructure nodes.** Giving all nodes publicly accessible IP addresses also means that the nodes can be accessed from everywhere in the world. This includes valid connection requests from trusted users and infrastructure services as well as malicious connection requests trying to compromise the node or gain sensitive data.

To allow multi-site applications to work in a secure and easy manner, direct access to all cluster nodes where the application runs is needed. Furthermore, to provide security, a dynamic, Grid-enabled firewalling strategy is needed. From the discussion above, the following requirements are derived:

1) Due to the fact that all virtual worker nodes are reachable from the Internet, the network security mechanism needs to ensure that while users can access their services on a cluster node, the node is guarded from both internal and external attacks.
2) The added flexibility and user empowerment (all user have root privileges inside their own VM) must not endanger other users or the infrastructure.
3) The network security configuration as well as the deployment must be easy, dynamic and scalable.

III. Related Work

Using a firewall as a Grid site protection mechanism raises several problems. For example, the Globus Toolkit 4 uses a wide range of ports for communication with Grid clients, thus it is a difficult task to configure firewalls properly [17].

One method to grant access to network resources is to use firewall traversal techniques. A system that uses such techniques is CODO [23]. CODO permits both inbound and outbound network traffic only for privileged applications. Apart from the fact that these decisions are made on the application level, the corresponding applications need to be linked with specific client libraries and require the use of firewall agents. A program's client library interacts with an agent to get the permission to traverse the firewall itself. However, the use of client libraries is unacceptable since the users of commercial software have no possibility to link the application to the required libraries.

Another idea is to use a semantic firewall [25]. Unfortunately, the applications need to present a profile describing their needs to the semantic firewall, such that an application has to be aware of the firewall's existence.

A dynamic firewall called *Dyna-fire* has been introduced by Green et al. [14] for a Globus Grid middleware environment. It supports VO-based security policies due to the modification of the Globus gatekeeper and enables fine-grained access control to Grid sites. The focus of Dyna-fire is the protection of Grid resources against attacks of unprivileged users (incoming traffic). In contrast, our solution restricts outgoing traffic as well, since we separate virtual execution environments from each other.

There are several proposals to create a secure container in which (untrusted) software can be executed. For example, Batheja and Parashar [3] use JavaSpaces to create a homogenous computing environment in a heterogeneous hardware landscape. The authors do not focus on security issues, and their approach is limited to Java applications. Since most of the cluster jobs are natively running Unix binaries, this approach is not suitable to satisfy our requirements.

Figueiredo at el. [9] presented a case study about Grid computing on virtual machines. They outlined the benefits to Grid computing by using operating system virtualization, especially in terms of security, usability and legacy support. Furthermore, they described several aspects of virtual networking including VPN access for clients to access their remote VMs.

Currently, Xen [2] is one of the most popular open source system in the area of virtualization. One of the main reasons is that virtual Xen machines, called DomUs, almost have the same performance as native cluster nodes due to the usage of para-virtualization [5], [2]. In contrast to full virtualization as used by some VMware products[1], para-virtualization offers a software interface to the virtual machine that is similar but not identical to the underlying physical hardware. Without hardware support of the CPUs [26], an operating system must be modified to run on top of Xen's virtual machine monitor (VMM).

Keahey et al. [15] have identified the need for integrating the advantages of virtual machines in the cluster and Grid area. It is argued that virtual machines offer the ability to instantiate an independently configured guest environment for different users on a dynamic basis. In addition to providing convenience to users, this arrangement can also increase resource utilization, since more flexible, but strongly enforceable

[1]`http://www.vmware.com`

sharing mechanisms can be put in place. The authors also identify that the ability to serialize and migrate the state of a virtual machine opens new opportunities for better load balancing and improved reliability that are not possible with traditional resources. Finally, security is increased by the use of virtual machines as compute environments [20], [21], [18].

The Globus project Virtual Workspaces [15], [16], [10] provides the dynamic creation of virtual machines in which Grid jobs are placed and executed. This provides the seclusion of Grid jobs from other jobs and running software, making it impossible for an unauthorized user to track the operating system activities during job execution. The Virtual Workspaces project is using the Xen virtualization technology for the dynamic creation and deployment of virtual machines. Since the latest version of Virtual Workspaces [11], users do not have to create virtual workspaces during job submission, but workspaces will be generated on demand. The virtual workspaces project is aiming at the Cloud usage pattern and is not integrated into a cluster scheduling environment and as such does not face and thus does not counter the threats discussed in this paper.

IV. Design

The basic idea is to use publicly accessible virtual worker nodes, guarded by transparent, dynamic firewalls.

A. Publicly Accessible Virtual Worker Nodes

As argued in Section II, service-oriented Grid applications require that public IP addresses are assigned to virtual worker nodes. An overview over the proposed architecture is shown in figure 1. A user submits a job a meta-scheduler instead of a site-specific headnode. The meta-scheduler handles the distribution of the job to a number of chose sites where an instance of the Globus Toolkit is installed. Every Globus handles the local scheduling in conjunction with the Grid Engine. At this point the Xen Grid Engines on all sites are invoked and coordinate the distribution of the Xen VMs to the worker nodes. The firewall solution, as proposed in the following sections, as well as inter-VM communication channels are set up. Once the VMs are ready on all sites job execution continues. Now the user is able to communicate with all assigned VMs.

In the following, we present our approach to allow such an application to use the described features in a secure and convenient manner.

B. Secure Infrastructure Communication

An important issue is to ensure a secure communication between the virtual worker nodes and the infrastructure services. This could be access to shared storage (accessible via NFS or Samba), automatic IP address configuration (DHCP) or host name resolution (DNS), as described below.

1) Access to Shared Storage: If the shared storage is only accessible from local, private IP addresses, devices with external IP addresses cannot access it. In the case where all virtual worker nodes have public IP addresses, there are two possible solutions:

- Only allow access from known IP addresses. This is the easiest solution and would be sufficient in most cases. However, this solution does not offer flexibility in case of address range changes, new nodes etc. Furthermore, it creates additional complexity in multi-site setups.
- Allow access on an application/job basis. Every application run has one or more VMs assigned with dynamically assigned IP address(es). Based on these addresses, access is granted on the same dynamic basis. This approach is more complex than the previous solution, but increases flexibility and scalability.
- Place all VMs and the storage server (commonly a NFS server) inside a dynamic, virtual VLAN. A virtual VLAN is the same as a physical VLAN, except that so called virtual switches are used instead of real ones. The VDE switches of the VirtualSquare Project [6] that tries to interconnect different virtualized entities can be used to achieve this goal. VirtualSquare even supports an encrypted virtual cable connection to interconnect the switches.

Due to the scalability benefits, the second solution was chosen. The solution guarantees that all VMs running on a particular cluster site only have access to their assigned storage.

Multi-site shared storage is not possible due to known problems with exporting filesystems over network borders. This drawback can be circumvented with either a dynamic Virtual Private Network (VPN) assigned between all participating sites. The possibility of a VPN between all nodes is discussed in subsection IV-D.

2) Grid Node IP Address Configuration: Every VM has a dynamically assigned IP address that is fixed during job duration. After the associated application finishes, the address is released. On each participating site, a DHCP server is setup to distribute the IP addresses. To prevent abuse of the service, DHCP requests are not allowed to pass through the routers guarding the network. Thus, all Grid nodes with a known MAC address can request an IP address. Nodes with unknown MAC addresses will not receive an IP address. Due to the fact that all users are superusers inside their VM, they could set a new MAC address and try to get another IP address from the DHCP. This is prevented by the presented solution by a fine-grained MAC address filter installed in domain 0 of the virtual worker node. The filter knows the legal MAC addresses of the running VMs and thus only allows DHCP requests if the valid MAC address is used. A properly chosen lease time ensures that no connection between the DHCP server and the VMs is necessary once the application started running. With this setup, we can guarantee automatic address configuration for nodes on all Grid sites.

One problem arises when using the presented setup: the management hassles introduced with multiple DHCP servers on several sites. To provide reliable IP addresses for the VMs, all DHCP server need to be synchronized. This could be accomplished by using a central resource managing the DHCP

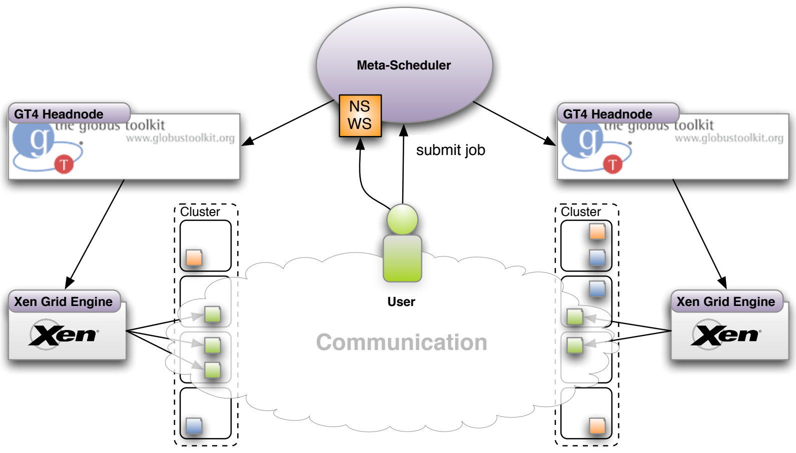


Fig. 1. Inter-Site communication between the user and the virtual worker nodes

configuration. This configuration is automatically distributed to all participating sites. Due to the fact that the number of VMs on one site is limited (the offered resources can only handle a certain number of VMs effectively), the distribution interval is not critical. It could run on a daily base.

If it is allowed to suspend or resume VMs while running, it could be possible that they could lose their DHCP lease and leave their personal firewall rules on the worker nodes behind. Currently suspending VMs is not supported by the XGE, so this is not possible. But this is a point the authors need to address in the future. One could think about a controlled suspend/resume initiated by the XGE. Thus it could take care of removing all firewall rules and bringing back the worker nodes to a coherent state. Once the VMs resume, they automatically apply for a new DHCP lease.

3) Host Name Resolution: Assigning host names to VMs is a necessary step to ensure identification for users and services (e.g. the resource manager, Globus Toolkit, monitoring software). Host name resolution is done by a DNS resolver installed on each Grid site. After a VM is started, we only allow connections to the well known DNS resolvers on the local site, connections to the DNS port on other machines are forbidden.

C. Dynamic Network Security

In the following, our novel design to guard the virtual worker nodes and to satisfy the stated challenges is presented.

1) Rule Set Generation: Network security and protection for the virtual worker nodes is enforced on domain 0 by dynamic firewall rules. To create firewall rules for a particular application belonging to a user, we use a default XML template. It represents a secure default setting for the average user, i.e. an uninterrupted workflow is possible without endangering other components. Based on this template, the user can apply his or her own rules. The basic template allows access to the LDAP, NFS and HTTP services by default and limits the network speed (this could be a default value or a value given by a SLA.)

2) Firewall Actions: To actually protect the network with the generated rules, the XGE has to deploy the script to the privileged domains 0 on the virtual worker nodes.

As described in subsection IV-D the XGE contains a list of VMs on which the application is executed. The list is obtained from the local Grid engine that decides on which hosts (VMs and corresponding domain 0s) the jobs will be executed. The generated firewall rules are now copied to all of these machines. Afterwards, the rules are deployed and become active. After the computation is finished, the XGE is also responsible for the removal of the deployed firewall rules. The XGE removes only the rules belonging to a certain application and user by managing a unique mapping between application, user and firewall rules.

It is indispensable to install the firewall rules on domain 0 of the host machine, not on the virtual machine itself. All users are superusers inside their VM, so they could easily remove the rules and thus bypass the protection. To enforce the protection offered by our rules, every connection that is not explicitly allowed is denied. In a starting state, a VM is not permitted to open outgoing connections. When the template rules are applied at first, connections to local services are allowed.

3) Traffic Limitations: To ensure that network connection bandwidth is not misused, the traffic of VMs can be limited. A default rate for all traffic is defined by the base template. Furthermore, it is possible to define limits for all types of connections based on a single port or port ranges, protocols or sets of hosts (e.g. the VM interconnection rate is higher than the connection to hosts outside this set).

4) Protection against Denial-of-Service attacks: Besides preventing bandwidth abuse from internal users, Denial-of-Service (DoS) protection is another threat the solution has to deal with. However, a complete and bullet-proof protection against this type of attack is not available. If only the attackers' bandwidth and the number of attack nodes is high enough, nearly every infrastructure could be spammed with unwanted traffic. To mitigate the DoS risk, some on-by-default countermeasures are put in place:

- Dynamic traffic limitation of ICMP messages to broadcast and multicast destinations from outside connections to prevent the so called *smurf* attack [4].
- A limitation of 100 incoming TCP SYN-Packets per seconds ensures a good SYN-Flood protection. The protection is bucked-based, so it does not affect performance if the number of SYN-Packets is under the given threshold.
- Incoming connections to ports that are likely to get scanned or abused (e.g. OpenSSH), are limited.

5) Information Exchange: To handle multi-site applications, a permanent communication channel between all running XGEs is needed. The purpose of the channel is to exchange information that needs to be present on all involved XGEs. Strong cryptography and authentication ensures that this information stays private and cannot be intercepted by malicious entities.

D. Inter-Node communication

Per default, the communication between the worker nodes is not encrypted. This is mainly due to overhead concerns. Encrypting the communication using a common cryptographic protocol produces CPU load, and the transfer rate is decreased. Nevertheless, setting up an encrypted connection between all worker nodes could be accomplished with a small overhead. All worker nodes are equipped with a SSH public/private key pair by default. This key pair is generated during creation time and is unique for every VM. After the VMs are deployed to the worker nodes, every VM contains the same key pair, so password-less access between all VMs is possible. To setup an encrypted tunnel OpenSSH can be used. Since Version 4.3 OpenSSH can be used to setup virtual private networks. With the built-in scripting support, a tunnel could be setup automatically during startup time or afterwards by issuing certain commands. The XGE could be modified to perform such a task.

V. Implementation

In this section, we describe our enhancements to the Grid nodes as well as to the XGE. Apart from the standard XGE features like VM handling and job management (which are described in detail in [22]), a network security module to implement the proposed design is introduced. The used infrastructure on every Grid site is based on a scenario presented in [19]. Every valid Grid user can use the Image Creation Station (ICS) to create his/her own customized virtual machines (VM). A user can log into his or her created VM and install and customize the VM to his or her needs. The VM is then deployed into an image pool. After a job is submitted by this user, the XGE chooses the user's own master VM image, generates several virtualized execution environments of the master image and boots them.

The first step after the XGE is invoked is to create a comprehensive set of rules. Based on a common template (see Section IV-C.1), a basic set of rules is created.

A. Deployment, Execution and Removal

Before the final rule set is deployed, it needs to be distributed to all participating XGEs that themselves distribute the rule set to the associated nodes. All XGEs are connected with each other with a SSL-secured communication channel. The final set of rules is copied to all participating virtual worker nodes (in fact to domain 0 that hosts the VM). The list containing the virtual worker nodes on all sites is available to the XGE (see Section IV-D). All rules are executed and thus the VM and the network are protected before the computation starts. After the computation is finished, all rules belonging to this job are flushed.

B. Packet Filtering

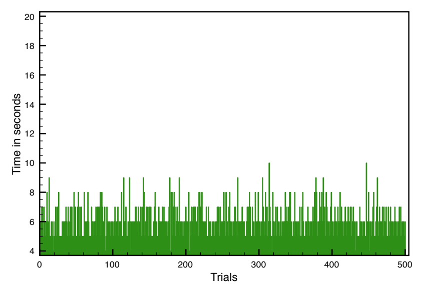
The Linux operating system is the most commonly used operating system in Grid/Cluster computing today. All of our nodes run Debian GNU/Linux and thus we use the *iptables*[2] packet filter. To match all packets originating from a VM, the *physdev*-module of the Linux kernel is used. This module allows filtering the traffic on the data link layer, which is needed due to the fact that Xen uses a bridge to connect the VM to the network.

VI. Experimental Results

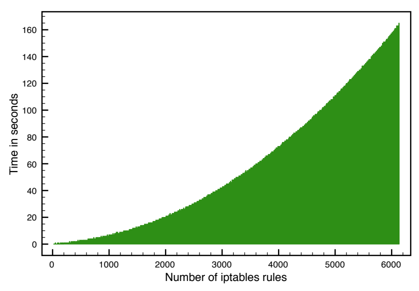
We conducted a number of measurements to investigate the properties of our proposal. The participating nodes are all AMD Dual-Core machines with 16 GB RAM running Debian GNU/Linux. The headnode also runs the Globus Toolkit 4.0.5 and the Xen Grid Engine. The native cluster nodes are interconnected with Gigabit Ethernet. All VMs have 2 GB RAM, one assigned CPU core and a local-loopback mounted hard disk image. The cross-site connection to other Grid sites is a 1 Gigabit link.

One of the important measurements is the time needed for rule propagation and final setup. The complete process includes rule generation for a particular application, determining all domain 0s belonging to that application, propagating the rules and set them up to protect the VM. The face detection application runs on 20 different VMs and 30 different rules protect the VMs. We conducted 500 trials to calculate a robust mean. The results are shown in figure 2(a). The mean of the results is 6 seconds and the deployment time is nearly constant over all trials. The small variance is due to changing load on the XGE node. The deployment time can increase if a larger number or more complex set of firewall rules is applied. This performance degradation is based on the performance of iptables. In figure 2(b) we measure the time needed to deploy an continuously increasing number of firewall rules with iptables. Applying a small number of rules, i.e. between one and 100 (which is the average case for most applications in our installation), this takes one second in average. Overall, the time needed for the installation increases logarithmically with the number of rules. As the rule setup is executed on all domain 0s in parallel, the additional cost is only counted once.

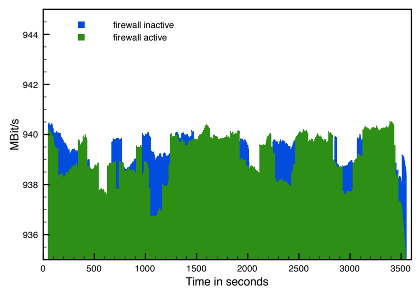
[2]http://www.netfilter.org/



(a) Dynamic firewall deployment time



(b) iptables rule deployment time



(c) iperf measurement w/o firewalls

Fig. 2. Firewall deployment measurements

Flushing the rules does not affect the performance at all. Even with a high number of rules, we were not able to measure a real impact.

Besides the time needed to install and deploy the firewall rules, there is also an impact involved when using firewalls at all. Measurements were taken between VMs on different physical hosts on two different academic locations interconnect via the German Research Network (DFN). As shown in Figure 2(c), there is no drawback in using firewalls shielding the virtual machines from each other, even when the machines are location on different sites. In total, 3600 samples were recorded during a one-hour-analysis of the network performance using *iperf*[3] as our measurement tool. The ordinate shows the network throughput in MBits per second, while the abscissa shows the time in seconds. When firewalls were used, an average network performance of 938.67 MBit/s was reached; when no firewalls were used, an average performance of 939 MBit/s was reached. A comparison measurement of the network performance between two physical machines using no firewalls shows a mean of 940.45 KBit/s. The slight decrease in network performance between virtual machines is due to the use of virtualization and the virtualized network stack. This measurement shows that applying an average number of rules will not affect the network performance.

For the sake of completeness a number of measurements were conducted with significantly more rules than are commonly needed for service-oriented applications. We conducted the set of measurements with a linear growing number of rules. The results are shown in figure 3. As the number of rules grows the maximum throughput is decreased. Overall the throughput is decreased about 80 MBits/s when using over 2000 rules.

In the next experiment, we performed measurements with the face detection application mentioned in Section II-A. The execution time of the service is not relevant to the firewall overhead and thus is factored out of the measurement, only the time needed to transfer the application data (enclosed in SOAP [27] messages) is measured. Figure 4 shows the result. The mean of the transfer time without a firewall (blue/left bar) is 79,9 s; the mean of the transfer time with a firewall (red/right bar) is 88,0 s. The resulting difference is about 8 s.

[3]`http://dast.nlanr.net/Projects/Iperf/`

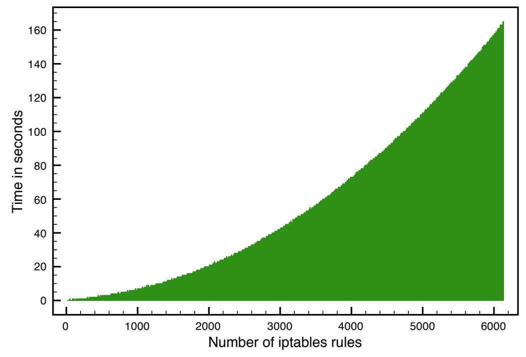


Fig. 3. Maximum throughput with an increasing number of firewall rules

Thus the measurement indicates that the use of firewalls helps to improve the security for a negligible small cost.

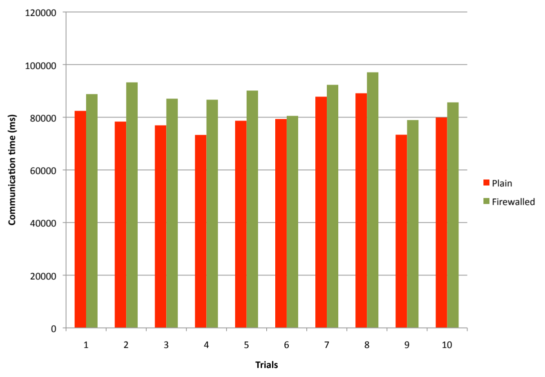


Fig. 4. Inter-application communication time (milliseconds) with and without dynamic firewalls

VII. Conclusions

In this paper, we presented a novel approach for secure multi-site virtual Grid computing. Dynamic and user adjustable firewalls protect the Grid infrastructure and shield users from each other. The firewalls are dynamically applied to the privileged Xen domain 0 of the cluster nodes to filter the traffic between the users' Xen virtual machines and the connected network. The decision to let users modify their own rule set in both directions and in a controlled manner is beneficial for both users and administrators. Running complex Grid applications requiring multi-site interaction is possible without bothering the Grid authorities to open ports on demand.

The threats identified in Section II-B are either negated or at least minimized. The system protects the Grid site infrastructure from being attacked by malicious users and secures the users' virtual execution environments against attacks by internal and external users. When an attack occurs to a virtual machine, concurrently running VMs are not affected. Inter-user networks attacks are prevented by default, since all connections between users' VMs are prohibited unless users explicitly allow the connections. The presented approach has been integrated into the Xen Grid Engine, a virtualization management software linked to the local resource manager. This step allows efficient generation as well as deployment of the firewall rule sets. Through this combination, the presented approach creates a secure-by-default environment, where security enables new functionality such as running fully service-oriented applications on existing Grid/cluster resources.

There are several areas for future research. A more extensive performance study with several hundreds of nodes is planned to improve the deployment mechanisms. Another interesting area for investigation is the optimization of the rule generation and propagation, as we have seen that this consumes some time. Furthermore, a self-adapting mechanism to control the Quality-of-Service parameters would increase the dynamic effect and reduce the administrator's involvement. Currently, a simple resource usage enforcement strategy is implemented. In the future, more complex systems such as the proposal by Feng et al. [8] could be adopted to enhance the present enforcement strategy. Finally, based on traffic analysis of the VMs, a service is planned to automatically increase or decrease the traffic limit on-the-fly.

ACKNOWLEDGMENTS

This work is partly supported by the German Ministry of Education and Research (BMBF) (D-Grid Initiative).

REFERENCES

[1] D-Grid Integrationsprojekt. http://www.d-grid.de/, May 2007.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 164–177, 2003.

[3] J. Batheja and M. Parashar. Adaptive Cluster Computing using JavaSpaces. In *Proceedings of the 3rd IEEE International Conference on Cluster Computing*, page 323, Washington, DC, USA, 2001. IEEE Computer Society.

[4] CERT Advisory CA-1998-01. Smurf IP Denial-of-Service Attacks. http://www.cert.org/advisories/CA-1998-01.html, 2008.

[5] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. Matthews. Xen and the Art of Repeated Research. *USENIX 2004 Annual Technical Conference*, pages 135–144, 2004.

[6] R. Davoli and M. Goldweber. Virtual square (v2) in computer science education. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, pages 301–305, 2005.

[7] R. Ewerth, M. Mühling, and B. Freisleben. Self-Supervised Learning of Face Appearances in TV Casts and Movies. *International Journal on Semantic Computing*, 1(2):185–204, 2007.

[8] J. Feng, G. Wasson, and M. Humphrey. Resource Usage Policy Expression and Enforcement in Grid Computing. *Grid Computing*, Jan 2007.

[9] R. J. Figueiredo, P. A. Dinda, and J. A. B. Fortes. A Case For Grid Computing On Virtual Machines. *23rd IEEE International Conference on Distributed Computing Systems*, 0:550, 2003.

[10] I. Foster, T. Freeman, K. Keahy, D. Scheftner, B. Sotomayer, and X. Zhang. Virtual Clusters for Grid Communities. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, pages 513–520, Washington, DC, USA, 2006. IEEE Computer Society.

[11] T. Freeman and K. Keahey. Flying Low: Simple Leases with Workspace Pilot. *Euro-Par*, page 10, Feb 2008.

[12] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a Virtual Machine-based Platform for Trusted Computing. *ACM SIGOPS Operating Systems Review*, 37(5):193–206, Jan 2003.

[13] T. Garfinkel and M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. *Proceedings of the 2003 Network and Distributed System Security Symposium*, pages 191—206, Jan 2003.

[14] M. Green, S. Gallo, and R. Miller. Grid-Enabled Virtual Organization Based Dynamic Firewall. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pages 208–216, 2004.

[15] K. Keahey, K. Doering, and I. Foster. From Sandbox to Playground: Dynamic Virtual Environments in the Grid. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pages 34–42, Washington, DC, USA, 2004. IEEE Computer Society.

[16] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. *Scientific Programming Journal 2006, Vol 13, No. 4*, pages 265–276, 2006.

[17] A. Rowland, M. Burns, J. Hajnal, D. Rueckert, and D. Hill. Using Grid Services From Behind A Firewall. Imperial College London, 2005.

[18] S. Santhanam, P. Elango, A. Arpaci-Dusseau, and M. Livny. Deploying Virtual Machines as Sandboxes for the Grid. *Proceedings of the 2nd conference on Real, Large Distributed Systems*, pages 7–12, 2005.

[19] M. Schmidt, M. Smith, N. Fallenbeck, H. Picht, and B. Freisleben. Building a Demilitarized Zone with Data Encryption for Grid Environments. In *GridNets '07: Proceedings of the First International Conference on Networks for Grid Applications*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[20] M. Smith, M. Engel, T. Friese, B. Freisleben, G. A. Koenig, and W. Yurcik. Security Issues in On-Demand Grid and Cluster Computing. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, page 24, Washington, DC, USA, 2006. IEEE Computer Society.

[21] M. Smith, T. Friese, M. Engel, and B. Freisleben. Countering Security Threats in Service-Oriented On-Demand Grid Computing Using Sandboxing and Trusted Computing Techniques. *Journal of Parallel and Distributed Computing*, 66(9):1189–1204, 2006.

[22] M. Smith, M. Schmidt, N. Fallenbeck, T. Doernemann, C. Schridde, and B. Freisleben. Secure On-demand Grid Computing. In *Journal of Future Generation Computer Systems*. Elsevier, 2008.

[23] S. Son, B. Allcock, and M. Livny. CODO: Firewall Traversal by Cooperative On-Demand Opening. In *Proceedings of the Fourteenth IEEE Symposium on High Performance Distributed Computing*, pages 233–242, Jul 2005.

[24] Sun Grid Engine Developers. Sun Grid Engine Website. http://gridengine.sunsource.net, March 2007.

[25] M. Surridge and C. Upstill. Grid Security: Lessons for Peer-to-Peer Systems. In *P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, pages 2–6, Washington, DC, USA, 2003. IEEE Computer Society.

[26] R. Uhlig, G. Neiger, D. Rodgers, A. L. Santoni, F. C. Martins, A. V. Anderson, S. M. Bennett, A. Kagi, F. H. Leung, and L. Smith. Intel Virtualization Technology. *IEEE Computer Journal*, 38(5):48–56, 2005.

[27] World Wide Web Consortium (W3C). W3C SOAP Specification. http://www.w3.org/TR/soap/.