Managing Critical Infrastructures through Virtual Network Communities

Fabrizio Baiardi[2], Gaspare Sala[1], and Daniele Sgandurra[1]

[1] Dipartimento di Informatica
[2] Polo G. Marconi, La Spezia
Università di Pisa
{baiardi,sala,daniele}@di.unipi.it

Abstract. Virtual Interacting Network CommunIty (Vinci) is an abstract architecture to share in a secure way an ICT infrastructure among several user communities, each with its own applications and security requirements. To each community, Vinci allocates a network of virtual machines (VMs) that is mapped onto the computational and communication resources of the infrastructure. Each network includes several kinds of VMs. Application VMs (APP-VMs) run applications and stores information shared within a community. File system VM (FS-VMs) store and protect files shared among communities by applying a combination of MAC and Multi-Level Security (MLS) policies. A firewall VM (FW-VM) is a further kind of VM that, according to the security policy of each community, protects information private to a community transmitted across an untrusted network or controls the information exchanged with other communities. The last kind of VM is the administrative VM (A-VM) that configures and manages the other VMs in a community as well as the resources of each physical node and it also assures the integrity of all the VMs.

After describing the overall Vinci architecture, we present and discuss the implementation and the performance of a first prototype.

Keywords: critical infrastructure, communities, virtual machines, trust level.

1 Introduction

Any complex ICT infrastructure is shared among distinct user communities, each with a trust level and proper security requirements. As an example, the ICT infrastructure of a hospital is shared among at least the doctor community, the nurse community and the administrative one. Each community manages its private information but it also shares some information with the other ones. As an example, users in doctor community can update the information about prescriptions while those in the nurse community can read but not update this information. The nurse community and the doctor one share other information with the administrative community that has to bill the patient insurances. In the most general case, each user belongs to several communities. Consider a doctor that is the head of a department. Being a doctor, she belongs to the doctor community but, because of her administrative duties, she belongs to an administrative community as well. To each community, proper rules and laws apply. As an example, in several countries information about the health of an individual should be encrypted when traveling

J. Lopez and B. Hämmerli (Eds.): CRITIS 2007, LNCS 5141, pp. 71–82, 2008.

on a public network to protect it from other communities. Any infrastructure, even critical ones, is shared among several communities. Hence, secure sharing is fundamental for the security of the overall infrastructure. First of all, this requires that each community should be able to define information it is willing to share with the other ones. The sharing can be implemented either as flows of information between two communities or through file systems shared among several communities.

To control information flowing between communities, a proper technology is the firewall one that has been defined to control the flows among networks with distinct security properties. This technology also supports virtual private networks (VPNs), to secure information transmitted across public networks. Instead, few tools are currently available [1] to support file sharing among communities. However, these tools can be composed to define the file system of interest.

We propose Virtual Interacting Network CommunIty (Vinci), an abstract architecture based upon VMs [2] [3] [4] to support the correct sharing of an infrastructure. A VM is an execution environment created by a virtualization technology that introduces a new layer into the computer architecture, the *virtual machine monitor* (VMM) [5]. This is a thin software layer in-between the OS layer and the hardware/firmware one that creates, manages and monitors the VMs that run on the VMM itself. In this way, the physical machine runs several VMs and the VMM confines the VMs so that any fault or error within a VM does not influence the other VMs. Since the VMM separates the VMs, any sharing of information is implemented through mechanisms that the VMs implement and control. To support several communities, Vinci assumes that each node of the ICT infrastructure runs a VMM and pairs each community with a virtual community network (VCN), i.e. a network of VMs. A Vinci VCN includes four kinds of VMs:

1. *Application VMs*, APP-VMs, that run the application processes of the community and store the community private information;
2. *File system VMs*, FS-VMs. Each FS-VM belongs to several VCNs and implements a file system shared among the communities paired with the VCNs it belongs to;
3. *Firewall VMs*, FW-VMs. A FW-VM controls some information flows to/from other communities, i.e. to/from the FW-VMs of distinct VCNs;
4. *Administrative VMs*, A-VMs. These VMs manage the configuration of the VCNs, and may also extend the VMM with a set of functionalities too complex to be implemented on top of the hardware/firmware level.

The VMs of all the VCNs are mapped onto the physical nodes with the goal, among others, of balancing the computational load of both the nodes and the interconnection network. To protect private information of a VCN that, after the mapping, is routed across low security links or nodes, A-VMs can create, configure and insert into a VCN further FW-VMs to implement a VPN among some VMs. In this way, the A-VMs and the FW-VMs extend the VMM to guarantee the separation among communities.

Furthermore, to increase the overall assurance, Vinci applies *virtual machine introspection* (VMI) [6] to detect attacks through consistency checks on data structures in the memory of the APP-VMs, and may run IDS tools on each VM.

The rest of the paper is organized as follows. Sect. 2 presents the overall architecture of Vinci and shows how it can be applied to a critical infrastructure. Sect. 3 discusses the current implementation and presents a first evaluation of the overall performance.

In the current prototype, FS-VMs and A-VMs run Security-Enhanced Linux (SELinux) [7] [8] [9] to support a large number of security policies that are defined and enforced in a centralized way. Sect. 4 discusses some related works. Finally, Sect. 5 draws a first set of conclusions and outlines future developments.

2 Vinci: Overall Architecture

We assume that the physical architecture of the infrastructure is a network spanning several physical locations and including a very large number of nodes. To manage the infrastructure in a secure way, we assume that each physical node runs a virtual machine monitor (VMM) that creates and manages a set of virtual machines (VMs). The VMM is responsible of the confinement among the VMs on the same node. We also assume that it guarantees fair access to the available resources.

For each community, Vinci introduces a distinct virtual network built through VMs belonging to the following classes:

- Application VMs (APP-VMs);
- Administrative VMs (A-VMs);
- File system VM (FS-VMs);
- Firewall VMs (FW-VMs).

The network that interconnects all the VMs related to the same community is seen as a Virtual Community Network (VCN). Vinci pairs each VCN with a *label* that denotes the security level of the community and includes mechanisms to enforce a set of common security policies to guarantee that a community can access the infrastructure and share information in a secure way.

Users of a community run applications on APP-VMs, which belong to one VCN and are paired with the same security label, the one of the corresponding community. An A-VM may either manage the configuration of the VMs of distinct VCNs on the same physical node or configure the VMs of just one VCN. Vinci redirects users to log on APP-VMs with the same label that identifies the user community. A FS-VM stores information shared among communities corresponding to the VCNs it belongs to and determines which of these communities can access any information it stores. A FW-VM has two roles: it rules the flows of information among distinct VCNs and protects a private flow of a VCN that is transmitted across low security networks.

2.1 Application VMs

Each APP-VMs has an associated minimal partition on one of the disks in the physical node. This partition stores the kernel of the OS that is loaded during the boot of the APP-VM. All the other files may be stored locally or in another VM of the same VCN or in a FS-VM shared with other communities.

Labels. During the boot process, APP-VMs are labeled with the security label that defines the community paired with the VCN they belong to. This label determines the kind of users that may log on these VMs. As an example, to manage the infrastructure of a company, the following communities can be defined:

- `R&D`;
- `engineering`;
- `sales`;
- `marketing`;
- `management`;
- `finance`;
- `customer support`;
- `services`.

For each community, a distinct label is introduced.

User IDs. The set of users of all the communities is globally known, so that users can be uniquely identified by their user-name or their associated UID. This global unique identifier, paired with each user, can be the same used by the OS of each VM to identify users, or a different one. In the first case, the same `PASSWD` file may be shared among all the VMs, because it stores the association between each user-name and the common global identifier paired with it. In the second case, the local UID is mapped onto the global one when accessing the resources, as an example by the FS-VM when serves a request.

IP address of a VM. Vinci statically assigns IP addresses to APP-VMs, and maps IP addresses into security labels so that each APP-VM inherits the labels paired with the IP address assigned to it. In this way, IP addresses uniquely identify the community paired with the VM. Since security labels depend upon IP addresses, proper checks are implemented to detect spoofed packets, so that all the requests to access a file can be authorized on the basis of the security label paired with the IP address of the APP-VM producing the request.

2.2 Administrative VM

Each A-VM has two roles, namely the configuration and the management of the VMs in a VCN and the interaction with the VMM of a physical node to configure the interaction environment. In principle, distinct A-VMs can be introduced for the two roles but, for efficiency consideration and easy of deployment, the tasks of all A-VMs mapped onto the same node are assigned to just one A-VM. This VM may also extend some of the VMM functionalities to simplify their implementation while minimizing the size of the VMM. Moreover, A-VMs in distinct nodes interact to manage the overall infrastructure. As an example, they may interact to determine or update the current mapping of a VCN.

The A-VM may also authenticate users in a centralized way, through a proper authentication protocol [10], so that users can log on APP-VMs of the corresponding community with the same combination of user-name and password. If several users can log on the same physical node, the A-VM may either direct users of the same community to the same APP-VM or create a distinct APP-VM for each user.

Each A-VM manages a private file system implemented on the disks of its physical node to create and configure the VMs of interest. As an example, the file system stores the kernel of the OSes that an APP-VM loads on start-up.

Sharing within a community. The files shared within a community include:

- Configuration files, such as those in `/etc`;
- System binaries, such as those in `/bin`, `/sbin`, `/usr/bin`, `/usr/sbin`;
- Shared libraries, such as those in `/lib`, `/usr/lib`;
- Log files, such as those in `/var/log`.

Other files are related to the applications of interest. To properly protect them, MAC policies may be adopted because, in general, no user of the APP-VMs need to update most of these files.

VM introspection. Vinci assigns to A-VMs the task of assuring the integrity of each VM mapped onto their node. To certify that each VM is in a good state, i.e. the kernel is not compromised, A-VMs apply VM introspection (VMI) [6]. VMI exploits at best the direct access of the VMM to the memory of each VM to gather information about a VM internal state. Starting from the raw values in the VM memory, the A-VM can rebuild the data structures of each VM, in particular of the OS kernel, and apply consistency checks to these data structure. As an example, the VMM can rebuild the process descriptor list of a VM to check that no illegal process is running. This is an example of how the A-VM can extend in a modular way the functionalities of the VMM.

2.3 File System VM

FS-VMs store files shared by users of several communities, i.e. each FS-VM stores files that can be accessed globally. These files include:

- User home directories. To minimize communication delays, these files can be mapped onto FS-VMs that run on nodes close to the user one;
- Projects and documents shared by a group of users.

These files are protected through a combination of MAC and MLS policy. As an example, MLS prevents an APP-VM with a lower security label from accessing files with higher security labels, whereas MAC policy enforces a standard security policy to all the users of all the communities. As an example, this policy may prevent append-only files from being overwritten in spite of the subject that invokes the operation. The FS-VM computes the security context paired with the subject of the MAC policy by mapping the UID and the label of the APP-VM requesting the operation into the security context.

2.4 Firewall VM

These VMs have two roles, namely to interact with an A-VM to protect private information of a community transmitted across an insecure network and to rule the flow of information between distinct communities. The first role is fundamental any time the VMs of a community have been mapped onto distinct physical nodes, i.e. they are remote VMs that are connected through a network with a lower security level than the one paired with the community and the VCN. In this case, the A-VMs on the considered nodes configure and activate proper FW-VMs to implement a VPN between the

VMs. FW-VMs intercept any communication among the remote VMs and encrypt and transmit the corresponding messages. Another role of the FW-VM is to filter information flows among distinct VCNs according to security rules. Here, a FW-VM of a community interact with the FW-VMs of other ones to filter information between the two communities. These FW-VMs are properly configured through the A-VMs of the considered communities as well.

Moreover, FW-VMs control that APP-VMs do not spoof traffic on the virtual bridge connecting the VMs on the same node. This guarantees that each request for a file can be authenticated, since the IP address of a VM is paired with the security label of the community and it is used to enforce the security policy.

2.5 Application to a Critical Infrastructure

Consider the critical information infrastructure controlling a public utility system such as that for the gas or water distribution. Among the communities that share this infrastructure, we have:

1. The *software community* that includes those that manages and update the software that the infrastructure runs,
2. The *SCADA community* that includes administrative users that access and update the parameters in the SCADA devices that control the distribution,
3. The *database community* that includes users that read some usage information on the SCADA devices to transfer it into a database used to bill the public utility users or to plan improvements and so on.

Each community shares some information with the other ones. As an example, the SCADA community uses the tools built by the software one. This sharing can be implemented through a file system that is updated and read by the software community, for example to configure the VMs used by the SCADA community. The SCADA and the database community need a shared access to SCADA devices. This can be implemented by considering these devices as private of the SCADA community and allow a flow of read requests from the database community to the devices. A FW-VM in the SCADA VCN receives this flow from the database community, checks the validity of the request and routes it to the VM that manages the proper devices. Information is then returned to the database community that stores it either in a private storage or in a FS-VM shared with other communities. Consider now user `Sally` that logs on the system from her physical machine. As we discussed previously, each machine runs a VMM that supports an A-VM that is responsible of the creation of APP-VMs on demand. Each APP-VM can find critical files such as system binaries, configuration files and shared libraries in its private file system. When the user logs on the system, the A-VM either logs the user to an existing APP-VM or creates a dedicated APP-VM and connects it to the proper VCN. If `Sally` belongs to several communities, the one to be considered may be either specified during the login phase or determined according to parameters such as the node the user is currently connected to, the time of the login and so on. The IP address assigned to the VM is known, and is statically paired with a security label. From now on, each request to access a file on a FS-VM is identified by the FS-VM according to the label of the APP-VM, such as `scada`, and the unique global identifier of the user,

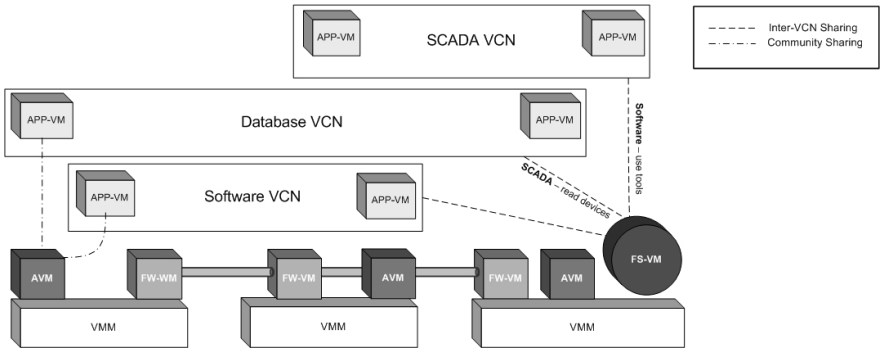

Fig. 1. Example

such as the UID associated with `Sally`. These two parameters are used to set up a SID for user `Sally` when requesting a file from a `scada` APP-VM.

3 Current Prototype

A first prototype of Vinci (see Fig. 2) has been implemented to evaluate both the feasibility of the abstract architecture and its efficiency and effectiveness.

3.1 Implementation

Xen [11] is the adopted technology to create the virtual environments that run the applications and export the shared file systems. To handle file requests and enforce the security policy, we used NFSv3 service [12] and SELinux. Both have been modified to apply security checks based upon the IP address of the requesting APP-VM and the UID. Finally, iptables [13] and OpenVPN [14] are used to handle the interconnection between the various VMs.

NFSv3 and SELinux Overview. The NFS service exploits a client-server architecture to implement a distributed file system by exporting one or more directories of the shared file systems to the APP-VMs. Every FS-VM and A-VM executes both a NFSv3 server and a SELinux module. SELinux implements MAC policies through a combination of type enforcement (TE), role-based access control (RBAC) and Identity-based Access Control (IBAC). The TE model assigns types to every OS objects, and the security policy can define the rules governing the interactions among OS objects. SELinux is based upon the Linux Security Modules (LSM) [15], a patch for the Linux Kernel that inserts both security fields into the data structures and calls to specific hooks on security-critical kernel operations to manage the security fields and to implement access control.

When the SELinux policy is being configured, every kernel component is labeled with a security context. At runtime, the security policy pairs each subject with its privileges to grant or deny access to system objects according to the requested operation. The integration of NFS with SELinux supports a centralized control of client access to the shared files and the assignment of distinct privileges to each APP-VM, leveraging the SELinux flexibility to describe MAC and MLS policies.

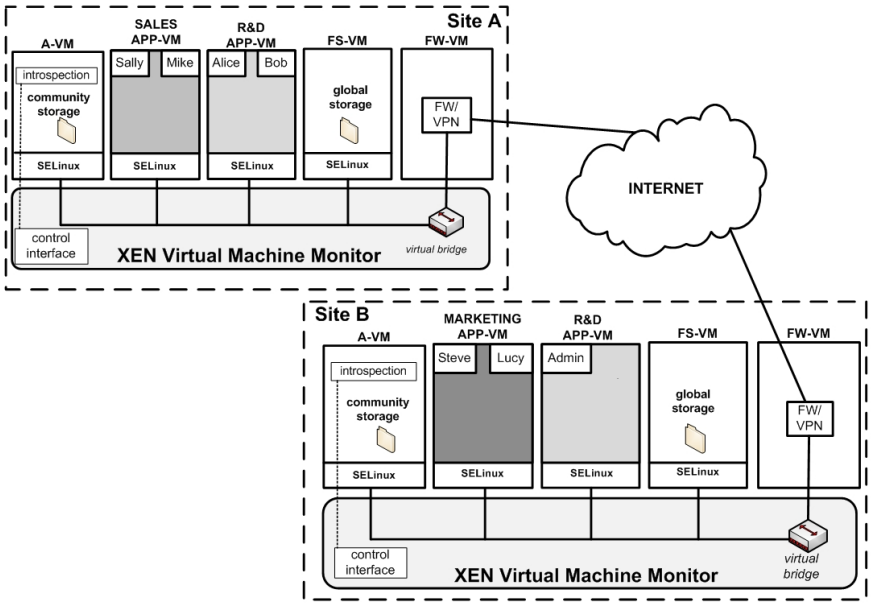

Fig. 2. Vinci Prototype

Interconnection. Since the FW-VM manages the interconnections among the VMs, a firewall determines whether two communities can interact. For this purpose, FW-VMs use iptables to: (i) decide whether to forward a packet, based on the source and destination community; (ii) to detect spoofed packets on the virtual bridge. In the first case, Vinci can be configured to isolate communities, so that communities with a lower security label cannot communicate with those with higher security labels. To protect information, FW-VMs on distinct physical nodes create a secure communication channel over the public network.

Assurance. In the current prototype, A-VMs use virtual machine introspection to discover any attempt to modify the kernel of every APP-VM running on the same VMM. To this purpose, an A-VM checks that:

1. Each APP-VM executes a known kernel image and that the kernel is not being subverted, i.e. no attacker is trying to install a rootkit to take control of the VM;
2. The list of the running modules in the kernel contains only certified modules, i.e. whose code signature is known and authenticated;
3. No APP-VM is sniffing traffic on the network.

These checks guarantee the integrity of the reference monitor of each APP-VM and they can be used if an attestation of the APP-VM software is required. This may be implemented by computing a proper hash value of the running software.

To guarantee the integrity of user applications running in APP-VMs, Vinci can delegate the implementation of security checks to the kernel running in the APP-VMs. As

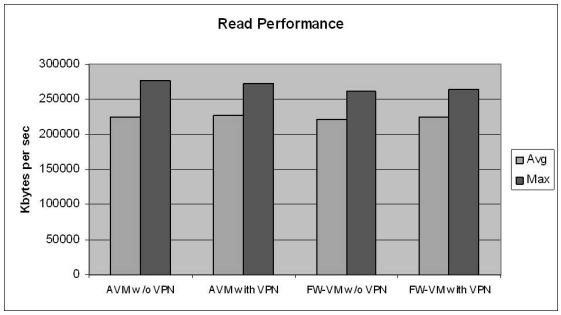

Fig. 3. Read Test

an example, we installed SELinux on APP-VMs to verify that no code is executed on the stack, to prevent a SUID executable to spawn a shell with root privileges. In this case, we use a *chain of trust* built from the VMM up to the higher layers, to apply and distribute the security checks at different levels. Therefore, each level controls the above one through tools that exploit the interface available at that level. In turn, this simplifies the development of the overall security mechanisms, by using appropriate tools at each level. Lastly, to detect spoofed packets Vinci defines an iptables `FORWARD` rule for every possible legal connection between two APP-VMs implemented by the virtual bridge of each VMM. Each rule is defined in terms of the static IP address of the virtual interface assigned to each APP-VM, that defines the community bound to the APP-VM. Every packet with a spoofed source IP address is dropped and logged. Since in the current prototype, the Xen privileged domain 0, the A-VM in Vinci, manages the virtual bridge, the task of detecting spoofed packets is delegated to the A-VM, rather than to the FW-VM.

3.2 Performance

We used the IOzone Filesystem Benchmark [16] to evaluate the performance of file sharing and of VPN. The benchmark has been executed on two nodes, one running

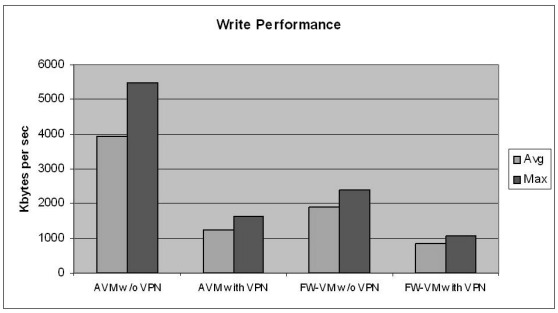

Fig. 4. Write Test

IOzone on an APP-VM while the other runs the FS-VM that stores the requested files. The nodes are connected through a 100MB Ethernet. During the test, we have used the Linux Debian distribution, Xen version `3.0.2`, NFSv3 and OpenVPN version `2.0.9`.

Four cases have been considered: with or without the FW-VM and with or without the VPN between the two nodes. Fig. 3 and 4 show, respectively, the average and max throughput of the IOzone `read` and `write` tests in each of the four situations.

4 Related Works

The notion of VCN derives from the one of virtual overlay used to describe peer-to-peer applications [17] [18]. As an example, a VCN can exploit its own routing strategy and use ad hoc algorithms to map information onto the VMs. [19] considers VM sandboxes to simplify the deployment of collaborative environments over wide-area networks. VM sandboxes are virtual appliances [20], configured and developed by the administrators of the collaborative environments, and made available to multiple users. This approach facilitates the joining of new nodes to the virtual network. [21] proposes an architecture where computing services can be offloaded into execution environments, *Trusted Virtual Domains* (TVDs), that demonstrably meet a set of security requirements. TVDs are an abstract union made by an *initiator* and one or more *responders* where, during the process of joining, all the parties specify and confirm the set of mutual requirements. During this process, each party is assured of the identity and integrity of the remote party's computer system. The enforcement of the attestation is delegated to virtual environments. *Labeled IPSec* [22] is a mechanism to enforce a MAC policy across multiple machines to control interaction among applications on distinct machines. [23] proposes an access control architecture that enables organizations to verify client integrity properties and establish trust into the client's policy enforcement. *Terra* [24] is an architecture for trusted computing that allows applications with distinct security requirements to run simultaneously on commodity hardware, using VMs. The software stack in each VM can be tailored to meet the security requirements of its applications. *sHype* [25] is a hypervisor security architecture that leverages hypervisor capability to isolate malicious OSes from accessing other VMs. This project is focused on controlled resource sharing among VMs according to formal policies. *Shamon* [26] is an architecture to securing distributed computation based on a shared reference monitor that enforces MAC policies across a distributed set of machines. *SVFS* [27] is an architecture that stores sensitive files on distinct VMs dedicated to data storage. Each access to sensitive files is mediated by SVFS that enforces access control policy, so that file protection cannot be bypassed even if a guest VM is compromised. Finally, *VM-FIT* [28] exploits virtualization to implement fault and intrusion tolerant network policies.

5 Conclusion and Future Developments

We believe that in the near future the use of virtualization technology will see widespread adoption inside organizations. This will facilitate the creation and deployment of virtual network communities each with users with the same requirements on security, service

availability and resource usage. The combination of VMMs, virtual machine introspection, firewall and a centralized enforcement of security policies for file sharing will guarantee: (i) the *confinement* of these virtual communities; (ii) the *integrity* of the VMs of a virtual community; (iii) a *controlled cooperation* among communities through the network or the shared files.

One of the future directions of our work is the use of attestation of the software on an APP-VM, before enabling the VM to join a virtual community. We plan to use virtual machine introspection as a technique to certify the software an APP-VM runs, and if the software is in a good state, i.e. it belongs to a list of allowed software and has not been compromised. The attestation can be dynamically re-sent to a certifying server, each time a new process is created inside the APP-VM or periodically, to guarantee the integrity of a VM.

Acknowledgments

We would like to thank Riccardo Leggio for his contribution to the prototype and the anonymous reviewers for their suggestions.

References

1. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 46–66. Springer, Heidelberg (2001)
2. User-mode Linux: The User-mode Linux Kernel Home Page, `http://user-mode-linux.sourceforge.net/`
3. VMware: VMware, `http://www.vmware.com/`
4. Xen: The Xen virtual machine monitor, `http://www.cl.cam.ac.uk/Research/SRG/netos/xen/`
5. Goldberg, R.P.: Survey of virtual machine research. IEEE Computer 7(6), 34–45 (1974)
6. Garfinkel, T., Rosenblum, M.: A virtual machine introspection based architecture for intrusion detection. In: Proc. Network and Distributed Systems Security Symposium (2003)
7. Enhanced Linux, S.: Security-Enhanced Linux, `http://www.nsa.gov/selinux/`
8. Loscocco, P., Smalley, S.: Integrating flexible support for security policies into the linux operating system. In: Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference, pp. 29–42. USENIX Association, Berkeley (2001)
9. Loscocco, P.A., Smalley, S.D.: Meeting critical security objectives with security enhanced linux. In: Proceedings of the 2001 Ottawa Linux Symposium (2001)
10. Neuman, C., Yu, T., Hartman, S., Raeburn, K.: The Kerberos Network Authentication Service (V5). RFC 4120 (Proposed Standard) (July 2005)
11. Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Pratt, I., Warfield, A., Barham, P., Neugebauer, R.: Xen and the art of virtualization. In: Proceedings of the ACM Symposium on Operating Systems Principles (October 2003)
12. Callaghan, B., Pawlowski, B., Staubach, P.: NFS Version 3 Protocol Specification. RFC 1813 (Informational) (June 1995)
13. Iptables: Netfilter/Iptables project, `http://www.netfilter.org/`
14. OpenVPN: OpenVPN - An Open Source SSL VPN Solution, `http://openvpn.net/`

15. Smalley, S., Vance, C., Salamon, W.: Implementing SELinux as a Linux security module. Nai labs report, NAI Labs (December 2001) (revised, May 2006)
16. IOzone: IOzone Filesystem Benchmark, `http://www.iozone.org/`
17. Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: ACM SIGCOMM 2001, San Diego, CA (2001)
18. Andersen, D.G., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient Overlay Networks. In: 18th ACM SOSP, Banff, Canada (October 2001)
19. Wolinsky, D.I., Agrawal, A., Boykin, P.O., Davis, J., Ganguly, A., Paramygin, V., Sheng, P., Figueiredo, R.J.: On the design of virtual machine sandboxes for distributed computing in wide area overlays of virtual workstations. In: First Workshop on Virtualization Technologies in Distributed Computing (VTDC) (November 2006)
20. Sapuntzakis, C., Brumley, D., Chandra, R., Zeldovich, N., Chow, J., Lam, M., Rosenblum, M.: Virtual appliances for deploying and maintaining software (2003)
21. Griffin, J., Jaeger, T., Perez, R., Sailer, R., van Doorn, L., Caceres, R.: Trusted Virtual Domains: Toward secure distributed services. In: Proc. of 1st IEEE Workshop on Hot Topics in System Dependability (HotDep) (2005)
22. Jaeger, T., Hallyn, S., Latten, J.: Leveraging IPSec for mandatory access control of linux network communications. Technical report, RC23642 (W0506-109), IBM (June 2005)
23. Sailer, R., Jaeger, T., Zhang, X., van Doorn, L.: Attestation-based policy enforcement for remote access. In: CCS 2004: Proceedings of the 11th ACM conference on Computer and communications security, pp. 308–317. ACM Press, New York (2004)
24. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: A virtual machine-based platform for trusted computing. In: Proceedings of the 19th Symposium on Operating System Principles (SOSP 2003) (October 2003)
25. Sailer, R., Valdez, E., Jaeger, T., Perez, R., van Doorn, L., Griffin, J.L., Berger, S.: sHype: A secure hypervisor approach to trusted virtualized systems. IBM Research Report (2005)
26. McCune, J.M., Jaeger, T., Berger, S., Caceres, R., Sailer, R.: Shamon: A system for distributed mandatory access control. In: ACSAC 2006: Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference, pp. 23–32. IEEE Computer Society, Los Alamitos (2006)
27. Zhao, X., Borders, K., Prakash, A.: Svgrid: a secure virtual environment for untrusted grid applications. In: MGC 2005: Proceedings of the 3rd international workshop on Middleware for grid computing, pp. 1–6. ACM Press, New York (2005)
28. Reiser, H.P., Kapitza, R.: VM-FIT: supporting intrusion tolerance with virtualisation technology. In: Proceedings of the 1st Workshop on Recent Advances on Intrusion-Tolerant Systems (in conjunction with Eurosys 2007), Lisbon, Portugal, March 23, 2007, pp. 18–22 (2007)