SECURING HEALTH INFORMATION INFRASTRUCTURES THROUGH OVERLAYS

Fabrizio Baiardi, Dario Maggiari
Polo G. Marconi - La Spezia, Università di Pisa, Italy

Daniele Sgandurra
Dipartimento di Informatica, Università di Pisa, Italy

Keywords: Security, security policy, healthcare infrastructure, virtual machine, overlay.

Abstract: Confidentiality and integrity of information are among the critical problems to face when managing health information through ICT systems. Virtual Interacting Network CommunIty (Vinci) is a software architecture that exploits virtualization to share a healthcare ICT infrastructure among users with different security levels and reliability requirements. Vinci introduces several *communities*, each consisting of users, some applications, a set of services and of shared resources. Users and applications with distinct privileges and trust levels belong to distinct communities. Each community is supported by a virtual network built by interconnecting virtual machines (VMs). The adoption of VMs increases the overall security level because we can use VMs not only to run user applications, but also to protect shared resources, control traffic among communities or discover malware. Further VMs manage the overall infrastructure and configure the VMs at start-up. Vinci supports the definition of security policies to protect information within and across communities. As an example, discretionary access control policies may protect files shared within a community, whereas mandatory, multilevel security policies may rule access to files shared among communities.
After describing Vinci architecture, we present the VM templates and preliminary performance results.

1 INTRODUCTION

It is widely recognized that health information technology can improve quality of assistance by increasing adherence to guidelines, enhancing disease surveillance and decreasing medication errors (Chaudhry et al., 2006), but it poses the problem of security and confidentiality of information. Some level of trust in security and safety of Information and Communication Technology (ICT) systems is fundamental to spread their adoption. We believe that several security problems of healthcare ICT systems can be solved through virtualization, a technology that replaces a physical machine with a software system, i.e. a virtual machine (VM), which emulates the physical machine. A VM can run any software that runs on a physical machine in a transparent way, i.e. without being modified. Virtualization supports the creation and the management of several VMs on the same physical machine through the introduction of a virtual machine monitor (VMM) (Goldberg, 1974). This is a thin software layer introduced in-between the OS and the hardware to support several VMs, and where each VM runs a distinct OS. The VMM is designed specifically to separate the VMs and to resist any attack from a malicious VM to another one. Important benefits of virtualization are the cost and the energy saving achieved by running several lightly loaded machines as VMs on a single machine (Uhlig et al., 2005). A further advantage is more robustness because virtual networks, i.e. networks of VMs, can also include VMs that check and control the other ones in a transparent way (Dunlap et al., 2002). In the following, a network of VMs is denoted as an *overlay* to stress that it is a logical entity built on top, and at most independent, of the physical network. Some VMs of an overlay run the applications and other ones monitor the overlay in an unobtrusive way. Since a VM is a software entity, the cost of a large number of VMs may be neglected with respect to that of further physical machines.

These considerations underlie the definition of

Virtual Interacting Network CommunIty (Vinci), a software architecture that exploits virtualization to share in a secure way a healthcare ICT infrastructure among users with different requirements and security levels. Vinci assumes that each physical node runs a VMM that multiplexes the node among several VMs and prevents a VM to access information of another one. The VMs are mapped onto the physical nodes according to community performance and reliability requirements and they can be migrated from a node to another one to better satisfy these requirements. Vinci introduces a distinct overlay for each *community*, and a further overlay to manage the infrastructure. A community is a set of users that access the infrastructure to execute applications, and of services that these applications exploit. A community is paired with a *global security level* that defines the set of users that can join the community, the applications they can run and the resources they can access. The adoption of Vinci simplifies the infrastructure management, because the VMs of an overlay can be homogeneously managed and protected since they have the same global security levels. This also simplifies the handling of security requirements, because VMs with conflicting requirements belongs to distinct overlays. To increase robustness against attacks, an overlay can also include VMs dedicated to monitor information flowing within and to/from other overlays. Distinct *VM templates* are defined to minimize the functionalities of each VM. Communities can cooperate and exchange information through VMs that belongs to distinct overlays. Obviously, security checks on information exchanged among communities are more severe than those within a community.

The rest of the paper is organized as follows. Section 2 presents the Vinci architecture and discusses the various VM templates to run user applications, monitor and manage the overlays. This section also presents preliminary performance results. Section 3 discusses a simple application of Vinci to a hospital infrastructure. Section 4 reviews some related works. Finally, Sect. 5 draws some conclusions.

2 VINCI OVERALL ARCHITECTURE

We assume that the infrastructure architecture is a private computer network that spans several locations, it includes a rather large number of physical nodes, and it is centrally managed by a set of administrators. Most of the infrastructure nodes are personal computers that are only accessed by one person at time and that some server nodes store shared data and execute server applications. Furthermore, each node runs a virtual machine monitor (VMM) to create and manage several concurrent virtual machines (VMs).

Vinci exploits the low cost of a VM to choose and properly configure its operating system (OS) according to the applications that the VM runs. This results in the definition of a set of highly specialized and simple VM templates that are dynamically instantiated and connected into overlays. Each overlay includes VMs that run applications and VMs that support and monitor the previous ones. While Vinci overlay may resemble a virtual private network (VPN), an important difference lies in the software complexity of a VM. As a matter of fact, to increase reliability and security, we minimize the number of applications that a VM runs and introduce VMs dedicated to apply consistency and security checks.

2.1 Templates, Communities and Overlays

To simplify the configuration of VMs and increase the robustness of each VM by minimizing the software it runs, Vinci introduces VM templates. In particular, the following templates are introduced:

1. *Application VM*: it runs a set of applications on behalf of a single user, i.e. a doctor or a nurse;
2. *Community VM*: it manages the private resources of a community by enforcing a community defined security policy;
3. *File System VM*: it is shared among overlays to protect information, i.e. health records, shared among the corresponding communities. It implements Mandatory Access Control (MAC) and Multi-Level Security (MLS) policies and a tainting mechanism to prevent illegal information flows;
4. *Communication and Control VM*: it implements and monitors private information flows, i.e. among VMs of the same community, or flows among communities, i.e. among VMs of distinct communities;
5. *Assurance VM*: it checks that Application VMs only run authorized software and that this software has not been attacked or affected by a virus;
6. *Infrastructure VM*: these VMs belong to a distinct overlay that manages the overall infrastructure.

When the VM is instantiated, the software environment that it supports is properly configured by choosing parameters such as the amount of memory, the OS and the VM applications according to the community of interest (Huang et al., 2006). This minimizes useless functionalities and increases the VM resiliency to malicious attacks.

A community is formally defined as a set of users and of applications having the same security and reliability requirements as expressed by the *community policy* that defines both the security level of users that can join a community and the applications that a user can run. This policy pairs a *global security level* with the Application VMs of the overlay supporting the community. Hence, a community may also be seen as built around a set of Application VMs with the same global security level. The number of communities reflects the distinct classes of users that may share an ICT infrastructure and the operations they can execute. As an example, when applying Vinci to a hospital ICT infrastructure, the following communities may be introduced: doctor, nurse, administration, accounting and help desk. However, to increase the overall security, the community a doctor belongs to when running a decision support system to access health records differs from the one she joins to surf the Internet. Vinci stresses the notion of a community as a collaborative environment where sharing of information among applications does not result in a loss of security or reliability. To achieve the security and reliability that a community requires, the corresponding overlay includes not only the Application VMs to run the applications of interest but also distinct VMs to implement and protect the information flows to/from other communities and the data shared either within a community or with other ones.

Application VMs. Each Application VM runs applications of a single user and is paired with the global security level inherited from the corresponding community. In general, this level constrains the resources and services the Application VM can access. As an example, Application VMs in the nurse community overlay can access and update the same information. Since a user can join distinct communities through distinct Application VMs, she can access distinct resources/services according to the overlay the corresponding VM belongs to. In Vinci, either a user chooses the community to join during the login phase or this community is automatically chosen according to the applications of interest. When adopting the second solution, communities can be transparent to users. Then, the Infrastructure VM of the local node configures and starts up one or several Application VMs to run the applications of interest and it inserts each Application VM into the proper overlay. In this way, on the same node, a user can concurrently run Application VMs that belong to distinct overlays.

In the current prototype, at boot time, an IP address is statically assigned to an Application VM. Both the global and the user security levels are paired with this address. Since the IP address uniquely determines the resources the VM and the user can access, Communication and Control VMs implement proper checks to detect any attempt of any VM to spoof traffic with the address of another VM.

Community VMs. A Vinci overlay always includes at least one Community VM to manage and control files shared within the corresponding community. In the current prototype, a Community VM protects its files through MAC and Discretionary Access Control (DAC) security policies where the subject of the policy is deduced from the IP address of the requesting Application VM. To this purpose, we have extended SELinux and Network File System (NFS) so that the NFS server enforces a policy where the subject is that paired with the IP address of the requesting Application VM. In more detail, SELinux labeling and access rules have been extended to introduce a distinct subject for each Application VM and to define the operations the subject can invoke. The object class of SELinux network object (*node*) has been extended to insert the operations that the server executes on behalf of the requesting Application VM, i.e. read, write, create file or directory. A node object is associated with the IP address of an Application VM to control the network traffic, i.e. to allow or deny the VM to exchange data with another VM.

File System VMs. Distinct communities can share information through a file system implemented by a File System VM that belongs to the corresponding overlays. The security policies that this VM enforces extend those of a Community VM by considering both the security of a VM and the community each user belongs to. Furthermore, File System VMs exploit a *Tainting module* to prevent the flowing of information among predefined communities. This module increases the robustness with respect to contamination attacks by confining information flows and by logging actual flows across communities. To this end, the Tainting module pairs each community and each file with a set of colours. A set represents the communities that either have interacted through the file or have exchanged information with the corresponding community. If a user of a community writes into a file, the set of the community is joined to the file set. Instead, if a user reads a file, the file set is joined with that of the user community. For each resulting set, some forbidden colours may be defined, i.e. the colours that cannot be paired with a file or a community. Anytime a user tries to apply an operation on a file, the Tainting module computes the resulting sets paired with, respectively, the file and the community. If at least one set includes a forbidden colour, then the operation is forbidden, otherwise it is executed and the sets paired with, respectively, the file and the community are updated. In any case, the Tainting module logs the operation type, the name of the community and of the file and the original and the new sets. Suppose, as an example, that two communities cannot exchange any information. In this case, we forbid the colour of one community in the set of the other one. Now, if a user *U* tries to read a file written by users in the other community, the operation is not executed because the resulting set of *U* would include a forbidden colour. Periodically, the Tainting module parses the log file and updates in an incremental way a dependency graph (King and Chen, 2005) that represents the information flows among communities and files. In the current prototype, the Tainting module has been inserted into the Linux kernel of File System VMs. By querying this module, an administrator can discover communities that have exchanged information, trace the source of an erroneous or malicious file update and track all the files/communities contaminated by an update.

Communication and Control VMs. A Communication and Control VM protects and monitors information flows by implementing and managing a local virtual switch that supports communications among VMs. Communication and Control VMs can be further specialized in: (i) *Secure Channel VMs,* which create an authenticated and protected communication channel among VMs of the same overlay that run on distinct physical nodes interconnected by a low security network; (ii) *Firewall VMs,* which filter information flows so that only authorized Application VMs exchange information with other communities; (iii) *IDS VMs,* which monitor information flowing in the same community or across distinct ones. As an example, a Secure Channel VM can be created when some connections among Application VMs are implemented through a wireless connection in a public space or through a network that cannot be trusted because it is outsourced. Firewall VMs can prevent communities with low global security levels to communicate with those with higher levels to avoid loss of confidentiality and to check that Application VMs do not generate fake traffic or attack other VMs. They also prevent a malicious user in a high security community to leak information to a low security community. Finally, IDS VMs monitor information flows in the same or in distinct communities to discover attacks from malicious users or external threats.

Assurance VMs. These VMs monitor the VMs of an overlay that manage critical components or run critical applications to attest that they have not been subverted by other VMs or by external threats. Assurance VMs exemplify the advantages of virtualization to build a robust system because they can use virtual machine introspection (Garfinkel and Rosenblum, 2003) to retrieve critical data structures in the memory of a monitored VM and check that neither the application nor the OS have been attacked by a worm or infected by a virus. As an example, an Assurance VM can discover that the configuration of a VM has been illegally updated or that it runs malware. These controls are fully transparent to the software in the monitored VM.

Infrastructure VMs. Infrastructure VMs configure and manage the overlays. They have been introduced to extend the functionalities of the VMM and minimize its size. In particular, they cooperate to monitor the overall infrastructure and update the mapping of VMs onto physical nodes. Any Infrastructure VM runs a minimal kernel, it does not run any Internet service and the functionalities it implements can only be directly accessed by the ICT infrastructure administrators.

Vinci introduces one Infrastructure VM per physical machine. All the Infrastructure VMs belong to a *Management overlay* that does not interact with any other overlay. Infrastructure VMs can: (i) create/kill, freeze/resume any VM in their node or request the operation to another Infrastructure VM; (ii) configure a VM through specific parameters according to the overlay topology, amount of memory, the number of VCPUs; (iii) retrieve information about the current VM mapping and the resulting resource usage; (iv) update the mapping by migrating VMs; (v) setup, compile and deliver to each File System and Community VM the general security policy it has to enforce. The Management Community migrate VMs among physical nodes to handle errors and faults, to reduce the communication latency or to balance the computational load. A migration requires an interaction among some Communication and Control VMs to manage the resulting communications topology.

If a high degree of assurance is required, some nodes of the infrastructure may be equipped with a Trusted Platform Module (TPM) subsystem (Pearson, 2002) to ensure that a platform has loaded its software properly and that both the VMM and the Infrastructure VM of the local node are started in a safe state. Moreover, the TPM can protect secrets such as the keys to encrypt traffic among VMs.

2.2 Performance Results

A preliminary security evaluation of the current prototype shows that Vinci can actually protect an infrastructure because, first of all, the adoption of Assurance VMs and introspection results in the detection of attacks that install in a VM malicious or dangerous software, such as a rootkit or a Trojan horse, before such a VM can harm the overall system. The average overhead due to virtualization is about 5%. In the current prototype, security checks introduce a further overhead that, in general, is lower than 13% of the execution time even in the case of complex security policies. It is worth stressing that our approach is fully transparent as it does not require to update applications or OSes so that any user can continue to use the applications she is familiar with.

3 AN EXAMPLE

Suppose that Vinci is applied to the ICT infrastructure of a hospital and that this infrastructure is shared among, at least, the doctor community, the nurse community and the administrative one. Each community is paired with a distinct overlay where some VMs manage the community private information while others ones store information shared among the communities. As an example, users in the doctor community can update information in a health record about prescriptions while those in the nurse community can read but not update it. Hence, information about prescriptions maybe stored in a database on a File System VM shared among the two overlays. This VM prevents the nurse community from updating the relations storing prescriptions. Furthermore, by pairing the prescriptions with a colour that is forbidden for any other community, the tainting mechanism can be applied to prevent prescriptions from being transmitted outside the two communities. Lastly, a Firewall VM can monitor the information flowing from the doctor community to the nurse one to prevent information leakage due to illegal information flow. The nurse and the doctor communities share other information with the administrative community that has to bill the patient insurances. Again, this can be implemented through File System VMs shared among the overlays. To simplify the handling of shared information, even a large number of File System VMs may be introduced because of their standard configuration and low cost. To show a case where a user belongs to distinct 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. Hence, it will use one VM to access administrative information and another one to access the health records of her patients. Furthermore, the doctor community can be further decomposed so that the community that a doctor joins to run a decision support system or to access confidential information about her patient health differs from that she joins when surfing the Internet. Both the VMs of the corresponding overlays run on a standard personal computer and the one that the doctor uses depends upon the application of interest. The ability of introducing several VMs at a very low cost makes it possible to select in a very detailed way the files that can be shared among communities and those to be encrypted.

Suppose now that the hospital allows the doctors to connect to the hospital infrastructure from their homes. This is possible provided that the home computer runs a VM that is configured by the hospital and that prevents other software, i.e. games or peer-to-peer software to be run. To further increase the overall security, the VM status can be remotely attested through TPM before allowing it to join an overlay and Secure Channel VMs can encrypt the information it exchanges with the hospital infrastructure. Obviously, users working from home should belong to a further community and the rights of users in this community, i.e. the operations they can execute and the data they can access, are a subset of those the same users have when connecting from their workplaces.

4 RELATED WORKS

(Griffin et al., 2005) proposes an architecture that offloads computing services into *Trusted Virtual Domains*, i.e. execution environments that meet a set of security requirements. *Trusted Grid Architecture* (Löhr et al., 2007) combines Trusted Computing and virtualization to build a trustworthy architecture where a user checks that the provider is in a trusted state before submitting a job. (Sailer et al., 2004) proposes an access control architecture where a Trusted Programming Modules verifies the client integrity before the client can access the corporate Intranet. *Poly*^{2} (Bryant et al., 2003) aims at segregating applications and networks and at minimizing OSes. It separates network services onto different systems and it isolates specific types of network traffic. Vinci shares with this framework the minimization of OSes and applications but it introduces distinct overlays for each community. (Wolinsky and et al., 2006) considers VMs as sandboxes that simplify the deployment of collaborative environments over wide-area networks. With respect to this framework, Vinci introduces communities to simplify the management of users with similar security and reliability requirements. PlanetLab (Chun et al., 2003) is a global overlay network that runs concurrently multiple services in *slices*, i.e. networks of VMs that include some amount of processing, memory, storage and network resources. While a slice recalls a Vinci overlay, Vinci allocates in a fairly static way resources of a private infrastructure, whereas in PlanetLab they are dynamically discovered and allocated in a world wide network. Furthermore, Vinci introduces communities to define flexible security policies and it exploits hardware-level virtualization rather than OS-level virtualization, to prevent a VM from accessing information of other VMs in the same node.

5 CONCLUSIONS

Vinci aims at a secure sharing of a healthcare ICT infrastructure among users with distinct security levels and reliability requirements. It assumes that each physical node runs a VMM to manage and protect the infrastructure resources. Vinci also defines several VM templates to support the execution of user applications, the enforcement of consistency checks, the implementation, the protection and the monitoring of information flows. VMs are connected into overlays, one for each user community. A community is defined according to the security and reliability requirements of its users and of their applications. A further overlay includes the VMs to create and configure the other overlays and map them onto physical nodes to achieve the required level of reliability and security. Preliminary performance and security evaluations show that Vinci can guarantee high security of a healthcare ICT infrastructure with an acceptable overhead. The overhead due to virtualization can be strongly reduced because multi-core architectures include a native support for multiplexing. Furthermore, they can assign a dedicated core to VMs that implement critical tasks, such as management and protection of other VMs, so that they are never delayed.

ACKNOWLEDGEMENTS

This work was supported by Promostudi and by Fondazione Cassa di Risparmio La Spezia.

REFERENCES

Bryant, E., Early, J., Gopalakrishna, R., Roth, G., Spafford, E., Watson, K., William, P., and Yost, S. (2003). Poly^2 Paradigm: A Secure Network Service Architecture. *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 342–351.

Chaudhry, B., Wang, J., Wu, S., Maglione, M., Mojica, W., Roth, E., Morton, S., and Shekelle, P. (2006). Systematic Review: Impact of Health Information Technology on Quality, Efficiency, and Costs of Medical Care. *Annals of Internal Medicine*, 144(10):742.

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. (2003). Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12.

Dunlap, G. W., King, S. T., Cinar, S., Basrai, M. A., and Chen, P. M. (2002). Revirt: enabling intrusion analysis through virtual-machine logging and replay. *SIGOPS Oper. Syst. Rev.*, 36(SI):211–224.

Garfinkel, T. and Rosenblum, M. (2003). A Virtual Machine Introspection Based Architecture for Intrusion Detection. *Proceedings of the 2003 Network and Distributed System Security Symposium (NDSS)*.

Goldberg, R. P. (1974). Survey of virtual machine research. *IEEE Computer*, 7(6):34–45.

Griffin, J., Jaeger, T., Perez, R., Sailer, R., van Doorn, L., and Caceres, R. (2005). Trusted Virtual Domains: Toward secure distributed services. *1st IEEE Workshop on Hot Topics in System Dependability*.

Huang, W., Abali, B., and Panda, D. (2006). A case for high performance computing with virtual machines. *Proc. of the 20th annual international conference on Supercomputing*, pages 125–134.

King, S. T. and Chen, P. M. (2005). Backtracking intrusions. *ACM Trans. Comput. Syst.*, 23(1):51–76.

Löhr, H., Ramasamy, H. V., Sadeghi, A.-R., Schulz, S., Schunter, M., and Stüble, C. (2007). Enhancing Grid Security Using Trusted Virtualization. In *ATC*, pages 372–384.

Pearson, S. (2002). Trusted Computing Platforms, the Next Security Solution. *Beaverton, USA: Trusted Computing Group Administration*.

Sailer, R., Jaeger, T., Zhang, X., and van Doorn, L. (2004). Attestation-based policy enforcement for remote access. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 308–317, New York, NY, USA. ACM Press.

Uhlig, R., Neiger, G., Rodgers, D., Santoni, A., Marting, F., Anderson, A., Bennett, S., Kagi, A., Leung, F., and Smith, L. (2005). Intel Virtualization Technology. *Computer*, 38(5):48–56.

Wolinsky, D. I. and et al. (2006). On the Design of Virtual Machine Sandboxes for Distributed Computing in Wide Area Overlays of Virtual Workstations. In *First Workshop on Virtualization Tech. in Distributed Computing (VTDC)*.