# Virtual Machine Contracts for Datacenter and Cloud Computing Environments

Jeanna Matthews
Clarkson University/VMware

Tal Garfinkel
Stanford University/VMware

Christofer Hoff
Packetfilter

Jeff Wheeler
Cisco

jnm@clarkson.edu, talg@cs.stanford.edu, choff@packetfilter.com, jewheele@cisco.com

## ABSTRACT

Virtualization is an important enabling technology for many large private datacenters and cloud computing environments. Virtual machines often have complex expectations of their runtime environment such as access to a particular network segment or storage system. Similarly, the runtime environment may have complex expectations of a virtual machine's behavior such as compliance with network access control criteria or limits on the type and quantity of network traffic generated by the virtual machine. Today, these diverse requirements are too often specified, communicated and managed with non-portable, site specific, loosely coupled, and out-of-band processes. We propose Virtual Machine Contracts (VMCs), a platform independent way of automating the communication and management of such requirements. We describe how VMCs can be expressed through additions to the Open Virtual Machine Format (OVF) standard and how they can be managed in a uniform way even across environments with heterogeneous elements for enforcement. We explore use cases for this approach and argue that it is an essential step towards automated control and management of virtual machines in large datacenters and cloud computing environments.

## Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection

## General Terms

Management, Security, Standardization

## Keywords

Virtualization, contracts, cloud computing, automation

## 1. INTRODUCTION

Virtualization is an important enabling technology for many large private datacenters and cloud computing environments. Much of the power of virtualization stems from the platform independence that virtual machines afford. The ability to snapshot a VM and run it elsewhere, the ability to deploy a generic virtual appliance that can be instantiated in an enterprise datacenter or a cloud computing environment, and many other examples all stem from this property. Unfortunately, today this property does not extend far beyond the limits of the individual virtual machines. While individual machines have been virtualized, many key elements for managing them in the aggregate have not.

For example, enabling the correct set of firewall rules for a virtual machine deployed in a datacenter would often require manual, out-of-band communication between many individuals. The designer of the virtual appliance knows the initial set of ports on which server software running inside the virtual machine is listening. The person deploying the virtual appliance knows the addresses of other servers with which the deployed virtual machine will be communicating. The administrators of the system/network into which the virtual machine is being deployed may have additional requirements to impose on the virtual machine such as limiting deployment of the VM to a specific isolated network segment. Information from all these individuals must be collected and communicated to the firewall administrator who actually applies the proper rules. Such manual, out-of-band processes introduce substantial opportunities for missing information and misconfiguration.

Further, such manual management practices may be completely impractical in an environment where deployment of new virtual machines is frequent and where there are no existing channels through which such person-to-person communication can flow. This is clearly the case in cloud computing environments and even in some large private datacenters. In such environments, VMs are deployed on demand and personal communication between end users and datacenter administrators is often restricted or non-existent (i.e. low "touch").

We propose Virtual Machine Contracts, a platform independent way of specifying, communicating and managing the complex expectations that virtual machines have of their runtime environment and vice versa. A Virtual Machine Contract (VMC) is a complex declarative specification of a simple question: Should a given virtual machine be allowed to operate in a specific runtime environment and if so, is that virtual machine currently operating within acceptable parameters?

In Section 2, we describe how VMCs can be expressed as an extension to Open Virtual Machine Format (OVF), a widely supported open standard for specifying, packaging and distributing virtual appliances. In Section 3, we discuss use cases for VMCs and describe why this approach is especially compelling in a large datacenter or cloud computing environment. In Section 4, we discuss how VMCs can be managed in a uniform way even across environments with heterogeneous elements for enforcement. In Section 5, we discuss the production or origin of information included in VMCs.

A Virtual Machine Contract is simple idea – adding management metadata to the package in which VMs are stored and communicated, but is also a powerful idea that is for the most part missing from the management landscape of large datacenters and cloud computing environments. We argue that standardization and support for VMCs is an essential step towards automated control and management of virtual machines in large datacenters and cloud computing environments. Unlike physical machines, VMs are digital objects to which contract metadata can be readily attached. Standardization and support for VMCs extends the platform independent nature of virtual machines themselves to the management of those VMs.

## 2. SPECIFYING VIRTUAL MACHINE CONTRACTS IN OPEN VIRTUAL MACHINE FORMAT

A Virtual Machine Contract is a complex declarative specification of a simple question, should this virtual machine be allowed to operate and if so, is it currently operating within acceptable parameters? If the answer is yes, then the VM should be allowed to operate, connect to the network, etc. -- if no, then appropriate action should be taken -- whether it is shutting down the VM, logging divergences from the policy or taking other remediation steps.

We propose that VMCs be expressed as extensions to Open Virtual Machine Format (OVF) [1]. OVF is hypervisor-neutral standard for describing, packaging and distributing virtual appliances. OVF is an industry standard developed by the Distributed Management Task Force (DMTF) with participation of representatives from many companies including VMware, IBM, XenSource, Cisco, Microsoft Dell and HP.

Requirements thus specified in the OVF package would directly accompany the virtual appliance through its lifecycle. Returning to our previous example of firewall policy, the correct set of rules could added to the contract at each step of the VM's lifecycle and then they could be communicated directly to the runtime environment when a VM is deployed, avoiding the manual communication and increased risk of misconfiguration.

An OVF package consists of an XML document called the OVF descriptor that specifies metadata about the virtual appliance and a set of additional content, typically virtual disk files. An OVF package can actually describe a set of virtual machines or virtual systems as they are called in OVF.

For each virtual system, the associated metadata is described in a set of specific sections. The *VirtualHardwareSection* describes the virtual hardware required including the amount of memory, number of CPUs, information about network interfaces, etc. The *OperatingSystemSection* describes the guest operating system that will run in the virtual system. The *ProductSection* provides basic information such as the name and vendor of the appliance and can also specify a set of properties that can be used to customize the appliance. The *EulaSection* describes the licensing terms for the virtual system and the *AnnotationSection* simply provides space for free form annotation.

In many ways, these sections already provide a type of Virtual Machine Contract. For example, the *VirtualHardwareSection* specifies aspects of what the virtual system requires from the runtime environment and the *OperatingSystemSection* describes aspects of what the runtime environment can expect from the virtual system once it is deployed.

We simply propose to extend OVF to include a much richer set of contract types. For example, a network contract could include a set of firewall rules specifying protocol, source/destination addresses and source/destination port. This information could be easily translated during deployment into the format required by software firewalls, hardware firewalls, network monitors, etc.

A concrete example is shown in Listing 1 of a proposed new section type, *ContractSection*. In this example, we specify that the virtual system, a web server, should accept incoming requests on port 80 and should initiate a connection to a database sever which is also specified in the same OVF package. Notice that instead of specifying IP addresses, which are assigned at runtime, the network contract in Listing 1 refers to VMs by ids defined within the OVF file (e.g. *ovf:id="webservervm"*). These ids can be mapped to IP addresses when a VM is deployed. This allows the contract to describe not just single VM behaviour, but expected patterns of communication between VMs.

Notice that the *ContractSection* in Listing 1 is marked as *ovf:required="true"*. This illustrates an important point. Contract sections could be marked as required or not. If they marked as required, then the hypervisor on which the VM is deployed is required to parse and enforce the contract or the VM cannot be deployed. If they are not marked as required, then enforcement of the section is optional.

Not requiring some sections would be a good way to phase in standardization and support for new contract elements. Unrequired sections would still be vastly superior to out-of-band communication or even to ad-hoc notations of information in the *AnnotationSection*, but would not prevent the deployment of a VM on a hypervisor that does not yet support the contract element.

Notice also that the *ContractSection* in Listing 1 has type *ovf: NetworkContract_Type*. This is meant to illustrate that some contract types could be specified in the OVF standard. Contract types that are already standard across a wide range of hardware and software products, like classic five-tuple firewall rules, are excellent candidates for initial standardization. As with any standardization effort, the set of contract elements that are widely supported and thus can be required would grow over time. Several authors are members of the DMTF group developing the next revision of the OVF standard.

**Listing 1. Network Contract Example**

```
<ns:ContractSection ovf:required="true"
           xsi:type="ovf:NetworkContract_Type">

     <Info> Network Contract for Webserver </Info>

     <Rule>
        <Info> Incoming web requests </Info>
        <Protocol> tcp </Protocol>
        <DstAddr ovf:id="webservervm" />
        <DstPort> 80 </DstPort>
        <SrcAddr> any </SrcAddr>
        <SrcPort> any </SrcPort>
     </Rule>

     <Rule>
        <Info> Connection to back-end DB </Info>
        <Protocol> tcp </Protocol>
        <DstAddr ovf:id="dbservervm" />
        <DstPort> 3306 </DstPort>
        <SrcAddr ovf:id="webservervm" />
        <SrcPort> any </SrcPort>
     </Rule>

     <Origin>
        <Info> Firewall protection for this VM is
required for regulatory compliance as it is
serving sensitive financial information </Info>
        <DateAdded> 2009-02-18 </DateAdded>
        <AddedBy name="John Foo" role="creator"</>
      </Origin>

 </ContractSection>
```

**Listing 2. Proprietary and Site-Specific Contract Examples**

```
<ns:ContractSection
xsi:type="cisco:IntrusionProtection_Type"
ovf:required="true">
     <Info> Site Specific IPS Policy </Info>
     <RequireCiscoIPS/>
     <Rule rule="43" setting="enabled" />
     <Rule rule="88" setting="disabled" />
</ContractSection>
<ns:ContractSection
xsi:type="local:NetworkAccess_Type"
ovf:required="false">
     <RequireLocalEngineeringNetworkAccess/>
</ContractSection>
```

Where the set of standardized contract types is insufficiently expressive to capture the important requirements, proprietary or site-specific contract sections could also be used. Listing 2 contains a concrete example. Proprietary or site-specific contract types that begin to gain wider acceptance would naturally serve as a starting point for further standardization.

## 3. USE CASES

In Section 2, we described how VMCs could be specified and communicated through additions to the Open Virtual Machine Format (OVF) Standard. In this section, we explore potential use cases that make this approach especially compelling for cloud computing or large datacenter environments.

***- Datacenter Automation and Management***

Certainly not all large datacenters or cloud computing environments use virtualization [4], but for those that do, VMC offer an opportunity to add rich metadata to VMs that can enable sophisticated life cycle management. Furthermore, VMCs specify what is required rather than how it is enforced opening the door to a mix of physical an virtual enforcement techniques depending on the needs of the specific datacenter environment. Managing the integration of virtualization specific infrastructure (e.g. virtual switches, virtual intrusion protection, virtual firewalls, virtual anti-virus protection, etc.) with traditional physical infrastructure is introducing even greater challenges [5, 6] which VMCs can help address.

***- Cloud Interoperability***

Many customers resist moving vital functions into any cloud for fear of vendor lock-in and availability problems from one carrier [2]. The more the full requirements of virtual machines can be expressed as standardized contract elements, the more customers will be able to use multiple cloud providers to assuage these concerns. We expect this "I won't play with you until you play nice with each other" attitude to ultimately drive the cloud and VMC standardization process.

***- Enterprise Datacenter to Cloud Migration***

The ability to run a workload in a VM and deploy it safely in a privately managed enterprise datacenter does not necessarily imply that it can be moved safely into a cloud (e.g. out from behind the local firewall, the local intrusion protection systems, etc.). VMCs can be used to automate the examination of criteria for moving a VM safely into a cloud environment. If the required contract section cannot be enforced in a new environment, this would correctly limit the migration of the VM to an external datacenter or cloud provider until appropriate action can be taken to address the problems.

***- Setting Bounds on Resource Consumption/Capacity Planning***

In any environment, assurance that a VM will be restricted to operating within certain bounds is valuable. However, for customers deploying VMs into an off-premise cloud offering, it is especially critical. In a cloud environment, the amount of resources available on-demand is much greater than on-premise infrastructure, substantially amplifying the potential impact of a VM acting in an undesired manner. One of the primary advantages of cloud computing to users is the ability to quickly expand their resource usage on demand. Given this, the administrators of a cloud environment could not assume that sudden increases in resource consumption are a sign of compromise. Thus resource limits expressed through VMCs could be quite helpful in bounding resource consumption and costs for cloud users.

Similarly, contracts can specify minimum and maximum amounts of a resource that a VM needs to function. This can also be an aid to capacity planning. Application level performance contracts. enabled by the software such as B-hive [3] where a VM could be required to provide some minimum level of service as a function of its resource utilization, could also help spot emergent problems.

***- Detecting Compromised Virtual Machines***

The specification and enforcement of contract rules help to turn virtual machines into appliances with a well-defined, limited purpose rather than general purpose computing devices. VMCs go along way towards delivering software that does only what is advertised and thus can help protect against virtual machines that become compromised. For example, the contract in Listing 1 specifies that the web server VM should only accept incoming connections on port 80. If the VM were compromised and attempted to open another port on which it could receive botnet command and control instructions, this unauthorized activity would be thwarted. Again, these types of limitations are useful in any environment, but are especially crucial in a cloud environment where the amount of resources available on-demand is much greater than in on-premise infrastructure, substantially amplifying the potential impact of a VM acting in an unadvertised, undesired manner.

***- Virtual Network Access Control***

Managing network access control policies is a key challenge in managing large datacenters of any kind. Any client which is running out-of-date software or which diverges from a standard site configuration represents a point of entry for an attacker. VMCs provide an easy platform for deploying virtual network access control (VNAC) policies that ensure that only correctly configured clients (and servers) can access the network.

If a system is not correctly configured, the VMC policy can allow it to run only in a mode that will permit remediation, e.g. connecting to a restricted VLAN to download patches, receive configuration updates, etc. There is certainly nothing new in the idea of doing NAC, rather VMC provides a means to realize this functionality as just one of its possible applications.

***- Regulatory Compliance***

Regulatory compliance is a large and growing concern for many businesses. HIPPA, Sarbanes-Oxley, and Payment Card Industry (PCI) regulations all have significant IT components. Virtually all the key elements for compliance could be touched on by a VMC such as requirements about host configuration (e.g. AV usage), network security (firewall use), security for data at rest (storage encryption) and data in flight (network encryption), and data retention (virtual rights management expiration policies), and network isolation (connectivity policies e.g.VLAN).

VMCs can helps an organization configure and audit its IT infrastructure to ensure compliance. For example, a contract could require that a VM containing sensitive data should only be powered on if it is on the proper protected network segment. The specification and enforcement of such requirements can help automate the negotiation of the safe migration of VMs from enterprise datacenter to cloud or from one cloud provider to another.

***- Disaster Recovery***

A key problem with disaster recovery is ensuring that a cluster at Site A will be able to fail over to the cluster at Site B. For virtual machines alone, ensuring this will work is a relatively simple matter as we can assume (modulo differences in processor type, etc.) that hardware we are recovering over to is uniform. However, the same cannot be assumed about the network, storage and other infrastructure elements with which a VM may expect to interact in its new environment. With VMCs, the list required infrastructure elements is known without needing to instantiate the VM instance on a recovery cluster.

## 4. ENFORCEMENT OF VIRTUAL MACHINE CONTRACTS

VMCs specify the requirements or policies that should be realized rather than how they should be enforced. Different datacenters may be running different hypervisors (e.g. VMware or Xen). As long as the contract is enforced, the details of enforcement are hidden from the virtual machine itself – extending the platform independent benefits of virtualization.

Similarly, different datacenters may deploy different methods for enforcement. For example, firewall rules like those specified in Listing 1 could be enforced by a virtual firewall implementation – either directly in the hypervisor or in a service VM such as a firewall VM through which all network traffic could be routed. However, firewall rules could also be enforced with a hardware firewall deployed upstream from the VM. In this case, a single hardware firewall could processes rules for many VMs running on multiple physical servers. Again, as long as the contract is enforced, the details of enforcement are hidden from the virtual machine itself.

An enforcement architecture is a service for connecting deployed VMs to enforcement elements that can support the rules contained in their contract sections. A simple prototype enforcement architecture has been demonstrated on VMware Virtual Center (VC) [7] with the following components:

- **Policy Language Processing** When a virtual appliance is deployed, rules are parsed from its OVF descriptor are placed in the VC inventory.

- **Rules Database** The VC inventory stores the contract sections associated with each virtual machine deployed in the datacenter.

- **Enforcement Element Notification** Currently, enforcement elements must poll the VC inventory for the addition of rules that they are configured to support. This prototype enforcement architecture does not recognize when there is no enforcement element available to support a required contract element. It would be better to enable enforcement elements to register as handlers for specific contract sections types so that an error could be raised when there is no registered handler for a given contract element.

A similar prototype could be implemented in any hypervisor supporting OVF. For example, a Xen-based prototype could be built using XenStore [8] to store the contract sections associated with each virtual machine and communicate them with registered enforcement elements.

The real work of enforcement is done in the enforcement elements, not in the enforcement architecture. The enforcement architecture provides a framework in which both existing and new enforcement elements can be deployed to support an ever-expanding list of contract types.

## 5. PRODUCTION OF VIRTUAL MACHINE CONTRACTS

In this section, we discuss how and why virtual machine contracts would be produced.

Rules could be added by the original designer of the virtual appliance to describe the expected behavior of the virtual machine. Rules could be added by the person deploying the VM or by anyone who modifies the VM. Rules could be added by the system or network administrator in charge of the environment in which the VM is deployed. Each of these individuals has knowledge of the VM's requirements that they are best suited to record and communicate.

The designer of a virtual appliance typically knows a great deal about its expected behavior simply based on the software they install and how they configure it [9,10,11,12,13]. They can specify when the appliance expects external resources, what hardware resources are required to provide a certain quality of service, what type of network traffic will be produced by the appliance and many other important pieces of information. Similarly, anyone that modifies the virtual appliance by adding or reconfiguring installed software could add similar information.

Contracts provide a way for appliance designers or modifiers to add value with information they already know. It also enables them to further differentiate the appliances they produce. In an appliance marketplace, they can compete not just based on the functionality and ease-of-use, but also on their ability to create an appliance with an appropriately, restrictive contract. Just like hardware appliances that consume less electricity, appliances that require less trust or fewer resources will be more attractive to users.

The person deploying a virtual appliance is best suited to know about the runtime environment and their intention for deploying the VM. For example, they know the precise identify of external resources to which they are connecting the VM. They may also know that a VM is being deployed for testing purposes only in a completely isolated testing environment or the expected load on a production service.

System or network administrators responsible for the deployment environment may also wish to add contract elements for VMs being deployed on resources under their control. For example, they may add network access control requirements such as maintaining a certain patch level for installed software or may set limits on the resources that can be consumed by the VM. Some such requirements could be appended automatically to the contact section of any VM deployed in their environment as a way of communicating global site requirements for all VMs.

VMCs enable the automation of communication between all the individuals involved in the life cycle of the virtual appliance. VMCs allow all such information to be packaged and distributed as a fundamental part of the virtual appliance itself.

In Listing 1, the example contract section includes information on the origin of the contract element. Information about who added the rule, when and why can help those deploying the VM later or in other environments to decide how to react if a desired runtime environment does not support the rule.

## 6. RELATED WORK

The Virtual Machine Contract was discussed in [14]. The focus in that work was on file system contracts that granted access to portions of user's personal data store with rich permission types such as “read-some” or “write-rarely”.

VMware's Application vServices are built-in services enforcement elements provided by a virtual datacenter operating system [15]. VMware vApps are collections of VMs specified in OVF. The OVF representation of a vApp can include a description of some operational policies and service level requirements.

## 7. FUTURE WORK

We have focused initially on the use cases for virtual machine contracts in large datacenter or cloud computing environments in part because the standardization process to be driven by two main factors – first, the need for cloud interoperability and the need to integrate virtual and physical versions of key infrastructure elements to manage large datacenters (e.g. virtual or physical switches, virtual or physical intrusion protection, etc.). However, VMCs can be extremely useful in the desktop and mobile environments as well. For example, the ability to detect and limit the impact of compromised VMs would be extremely beneficial in desktop environments, especially home machines with high bandwidth, always-on Internet connections that are the favorite targets of botnets. Similarly, as virtualization moves into the hand-held computing environment, VMCs can be used to negotiate the highly varied space of hardware characteristics (e.g. to determine if a given runtime environment has sufficient display features to support a given VM.)

We are working actively on standardization of virtual machine contracts. Several authors are members of the DMTF group developing the next revision of the OVF standard. The benefits of VMCs require building wide spread support for recording and enforcing VM requirements in a standardized format.

Finally, we described contracts as a declarative specification, but an enforcement architecture could also contain a scripting engine to support imperative contract rules. Such a scripting engine would enable periodic execution of scripts written to assess compliance with a given rule (e.g. causing a request to be sent to a blocked port to determine if it is indeed blocked).

## 8. CONCLUSION

Virtual Machine Contracts are a simple idea but also crucial step towards automated control and management of large datacenters and cloud computing environments. VMCs help enable VMs to can migrate safely and naturally within a datacenter, between on and off premise capacity and between multiple cloud providers. They help bound the resources consumed by VMs and detect/thwart VMs that become compromised. They can help manage thorny issues such as virtual network access control and regulatory compliance. It is our sincere desire that a discussion of VMCs in this workshop on automated control for datacenters and controls can help to build interest and demand in the community for the deployment and standardization of rich management metadata associated with VMs.

## 9. ACKNOWLEDGMENTS

Thanks to Winston Bumpus, Larry Lamers, Orran Krieger, Stephen Evanchik, Rene Schmidt, Steffen Grarup, Bich Le, A.B. Srinivasan, Warren Wu, Wendy He and Christian Dickmann at VMware for feedback on these ideas.

## 10. REFERENCES

[1] DMTF System Virtualization, Partitioning and Clustering Working Group, "Open Virtualization Format (OVF) Specification, Version 0.90", April 2008.

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing", University of California at Berkeley Technical Report No. UCB/EECS-209-28, February 10 2009.

[3] B-hive, Application Performance Management for VMware Infrastructure, http://www.bhive.net/.

[4] J. Fry, "Sorry, VMware: you don't need virtualization for cloud computing", http://datacenterdialog.blogspot.com/2009/02/sorry-vmware-you-dont-need.html, February 25 2009.

[5] VMsafe: A Security Technology for Virtualized Environments, http://www.vmware.com/overview/security/vmsafe.

[6] C. Hoff, "The Four Horsemen Of the Virtualization Security Apocalypse", Black Hat 2008.

[7] VMware Virtual Center, http://www.vmware.com/products/vi/vc/.

[8] XenStore, http://wiki.xensource.com/xenwiki/XenStore.

[9] VMware Virtual Appliance Marketplace, http://www.vmware.com/appliances.

[10] Bagvapp Virtual Appliance Repository, http://bagside.com/bagvapp.

[11] RPath, http://www.rpath.com.

[12] Thoughtpolice VMware Images, http://www.thoughtpolice.co.uk.

[13] Jailtime, http://www.jailtime.org.

[14] J. Matthews, J. Herne, T. Deshane, P. Jablonski, L. Cherian, M. McCabe, "Data Protection and Rapid Recovery From Attack With A Virtual Private File Server and Virtual Machine Appliances", Proceedings of the IASTED International Conference on Communication, Network and Information Security (CNIS 2005), p. 170-181, November 2005.

[15] VMware Application vServices, http://www.vmware.com/technology/virtual-datacenter-os/application.html.