Configuration Management Security in Data Center Environments

Krishna Kant

George Mason University
kkant@gmu.edu

Abstract. Modern data centers need to manage complex, multi-level hardware and software infrastructures in order to provide a wide array of services flexibly and reliably. The emerging trends of virtualization and outsourcing further increase the scale and complexity of this management. In this chapter, we focus on the configuration management issues and expose a variety of attack and misconfiguration scenarios, and discuss some approaches to making configuration management more robust. We also discuss a number of challenges in identifying the vulnerabilities in configurations, handling configuration management in the emerging cloud computing environments, and in hardening the configurations against hacker attacks.

1 Introduction

As the size and complexity of data centers grows, so does the sophistication and complexity of managing its myriad resources including computational elements, storage systems, networking fabrics, software and services. The hardware and software infrastructure must be managed at multiple levels spanning from individual devices (and associated device drivers) all the way up to the entire data center infrastructure and operations. The management is needed from multiple perspectives (e.g., performance, availability, security, energy efficiency, etc.) and over the entire *life cycle* of the hardware and software. This is the scope of what is loosely referred to as "configuration management" (CM) and it involves a large variety of data obtained and stored in various ways. It is clear that such data must be protected from unauthorized access and corruption.

The Operating System, middleware, and applications often maintain configuration information in a rather primitive form, i.e., in "configuration files", which can be easily misconfigured or corrupted. However, the management planes of enterprise systems use a much more organized hiearchay of databases for keeping and manipulating CM data. In the past, CM dealt mostly with configuration of physical equipment and was done in out-of-band (OOB) manner using specially designed management software running a separate processors. The resulting CM repositories were therefore relatively isolated and available only to the physically separate management infrastructure; however, the sophisticated management requires a close coordination between OOB and normal in-band management, which makes the OOB management much more vulernable.

The recent trend of widespread virtualization significantly complicates the CM since the virtualization effectively turns "hard" assets such as a server with fixed set of resources into "soft" assets with dynamically varying parameters. For example, the set of hardware threads, physical memory, or network interface bandwidth dedicated to a virtual machine (VM) could be changed dynamically. The file based VM configuration is highly vulnerable to misconfigurations and attacks, but recent efforts to standardize VM configuration is helpful in this regard (see open virtualization format [11]). The emerging trends of outsourcing and multi-tenacy further increase the complexity and vulnerability due to configuration information flow related restrictions and lack of trust. These attributes provide new avenues for attack on CM and increase chances of misconfigurations.

In this chapter, we expose CM vulnerabilities and discuss challenges in making it secure. We also propose some mechanisms that include exploitation of the special structure of data centers and discuss diversity techniques for hardening the defenses against attacks. Although a substantial amount of literature exists on protecting specific aspects of a computer networks and systems (e.g., routing tables, databases, etc.), the research on protecting configuration management systems per se has been quite limited. We shall point out related work as appropriate in subsequent sections.

The outline of the chapter is as follows. Section 2 provides an overview of the configuration management infrastructure and standards in enterprise systems. Section 3 discusses in detail the consequences of attacks on configuration management data and the mechanisms that hackers can exploit for attacking it. Section 4 discusses some general security mechanisms for protecting CM data, including those that exploit the redundancy that naturally exists in data centers. Section 5 then discusses the challenges in securing configuration management both in traditional environments and in emerging cloud computing environments. Finally, section 6 concludes the discussion.

2 Configuration Management Basics

Configuration management refers to all aspects of ensuring that the data center continues to provide the desired services. It involves continuous tracking, update, and troubleshooting of HW, SW and service configurations. In this section, we discuss the evolution and current status of configuration management in enterprises.

2.1 Scope of Configuration Management

Systematic configuration management has its roots in telecommunications systems, which follow the FCAPS model specified in ITUs telecommunications network management (TNM) standards [15]. FCAPS enumerates five categories of management activities: Fault, Configuration, Availability, Performance, and Security. Although "Configuration" is only one of these, all types of management

must deal with substantial amounts of data that can be loosely defined as configuration. For example, security involves a variety of configuration data such as firewall rules, authentication methods, parameters and keys, access control setup and rules, etc. In fact, configuration management is central to ensuring desirable properties for the system, be it performance, availability, security, etc. With energy (and correspondingly power and thermal) management becoming crucial in data centers, the corresponding configuration data (e.g., power states at various levels, temperature limits, airflow requirements, etc.) also needs to be maintained. We can therefore refer to an expanded FCAPSE model for modern data centers, where the "E" stands for energy.

Although "configuration" conceptually refers to system aspects that are relatively static, such a notion of configuration is inadequate in modern data centers. First, with wide-spread use of virtualization, the configuration needs to include virtual machine settings, which can change dynamically and could migrate from server to server. Second, a flexible resource management often implies that the relevant policies and parameters are explicitly specified and manipulated instead of being buried in the management code. This applies not only to the physical infrastructure and its abstractions but also to software and services as well. Consequently, configuration management becomes a truly central aspect of modern data centers.

Another aspect of configuration management is its temporal span, i.e., keeping track of assets over their entire life-span. In particular, a "hard" asset such as a server or switch, needs to be managed from the moment it is brought into the data center until it is removed (due to failure or retirement). The various stages along this temporal scale include: (a) automated discovery of capabilities and "qualification" of the asset (i.e., checking authenticity of all already installed hardware, firmware and software), (b) configuration and provisioning of the asset with desired OS and utilities to make it available for normal use, (c) provisioning of services (and their reprovisioning as conditions/needs change), (d) active performance and availability monitoring and tuning, (e) diagnosis, trouble-shooting, reconfiguration, repair, upgrade, etc. as needed, and (f) eventual removal and retirement of the asset. Fig. 1 pictorially shows the management attributes and life-cycle management of an asset. The life-cycle management applies to software as well since new software must be authenticated and properly configured, existing software frequently patched ("repaired" or "upgraded"), and ultimately the software must be retired (although the virtualization technology allows older and newer software versions to continue running side by side). It is also important to note that increasingly, sophisticated software and its management is crucial not only for the servers but also for other assets including switches, storage bricks, security appliances, etc.

The organization of data centers naturally defines a hierarchy which can be exploited for configuration management. Servers are typically placed in a fixed size rack, either directly or – in case of blade servers – in a chassis that fits the rack. A chassis often has sophisticated built-in management capabilities for the blades hosted in it. Increasingly, racks also sport management capabilities for the

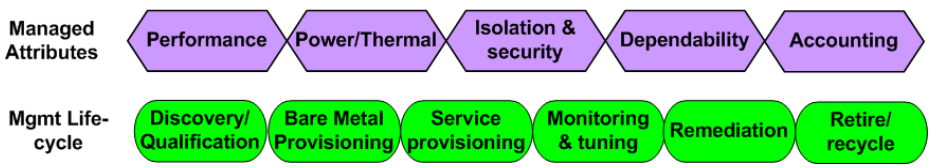

Fig. 1. Illustration of Management Attributes, Life-Cycle and Activities

assets installed there. The next level may be a cluster of racks and ultimately the data center level. Such a hierarchical structure simplifies management and allows for scalability of the physical infrastructure. The network – both mainstream as well as storage – also follows a similar fat-tree structure and hence can be managed in a hierarchical fashion [22]. Even in a virtualized environment, the hierarchical structure is useful in allocating all VMs for an application physically close together.

2.2 Configuration Management Infrastructure

Configuration management involves a variety of repositories that generally follow the hierarchical system structure. In particular, at the lowest HW levels, each device carries a firmware repository containing both fixed and settable parameters and their current values. At the next level, a subsystem (e.g., control plane of the router) or system (e.g., entire server) will have its own firmware or SW repository containing the appropriate parameters (e.g., amount of memory installed). The higher level parameters may or may not be related to lower level parameters simply. For example, the internal BW of the switch is usually less than the sum of individual bandwidths supported by all the ports.

Configuration repositories generally follow the standard Common Information Model (CIM) developed by the Distributed Management Task Force (DMTF) [7]. CIM is a hierarchical modeling language based on UML (unified modeling language) for defining objects and relationships between them. These relationships could be structural or more abstract, e.g., binding between virtual machines (VMs) and the physical server they run on. CIM unifies and extends existing instrumentation and management standards (e.g., SNMP, DMI, CMIP) by providing both schemas and a specification language. A CIM schema defines an object in the entity-relationship style and allows for compact representations of complex objects using concepts such as nested classes, instantiation, inheritance, and aggregation. For example, a CIM model of a switch includes its physical structure, various parameters required for configuring it (e.g. per port and shared buffer, packet formats supported, port speeds, etc.), their current settings, and methods to change the values.

DMTF has also developed Web-Based Enterprise Management (WBEM) specification that provides mechanisms to exchange CIM information in an interoperable and efficient manner. CIMOM (CIM object manager) is an open-source implementation of WBEM. The components of WBEM include access to

CIM information in a variety of ways including web services based management (WSMAN), CIM query language, CIM command language interface (CLI), etc. However, WSMAN is becoming quite popular and runs atop SOAP (simple object access protocol), which itself is layered on top of HTTP. WSMAN consists of several components: (a) WS-Addressing – defines references to web service endpoints, (b) WS-Transfer – implements basic access functionality such as get, put, create, delete, (c) WS-Enumeration – allows iteration through members of a collection, (d) WS-Eventing – supports publish/subscribe interface, and (e) WS-Security – provides authentication and encryption services for SOAP communications (discussed later). The security procedures are optional and often bypassed due to performance reasons. This allows a wide variety of web-services based attacks on CIM repositories, as discussed in section 3.2.

CIM based models can be used for representing the configuration of entities beyond individual HW assets, such as configuration data for a rack or chassis hosting a number of servers, switch, storage boxes, etc. The configuration data in this case could involve rules for allocating power among the assets, fan control parameter, establishing share or uplink bandwidth among servers, etc. Similarly, CIM models can be specified for specifying the configuration of individual VMs or a network of VM's forming a "virtual cluster". In fact, DMTF has defined a standard CIM/XML representation of VM's called Open Virtual Format (OVF) [11]. Such a format is substantial improvement over "configuration file" based representation in that it is standard, vendor-independent and admits storage and maniupulation using above-mentioned technologies.

DMTF also has some ongoing initiatives for implementing vendor and OS independent management of assets. This includes SMASH (Systems Management Architecture for Server Hardware) for managing HW assets in pre-boot state and CDM (Common Diagnostic Model) for diagnostics [12]. The Virtual Management Initiative (VMAN) provides a comprehensive OS independent VM management (creation, allocation, monitoring, etc.) capability to manage VM's (represented using OVF) that can be invoked by the OS and middleware [10]. In spite of these initiatives, the need for rich data management capabilities are often provided by vendor-specific and domain specific management packages. For example, one package may be used for provisioning OS on bare machines, while another one is used for performance monitoring. Invariably, such vendor-specific management packages (VSMPs) include their own private configuration databases that maintain relevant information in a way that they deem most suitable. The maintenance of potentially overlapping information in different databases in different formats can lead to many difficulties including misconfigurations and exploitations by malicious users.

The configuration data in a data center naturally follows a hierarchy with one or more top level configuration management databases (CMDBs) that consolidate information from all repositories. Figure 2 illustrates this with 2 CMDB's at the top. CMDB's typically support the federated model which can allow them to scale to large installations. CMDBs are important to capture dependencies between various packages and HW/SW components and provide a global

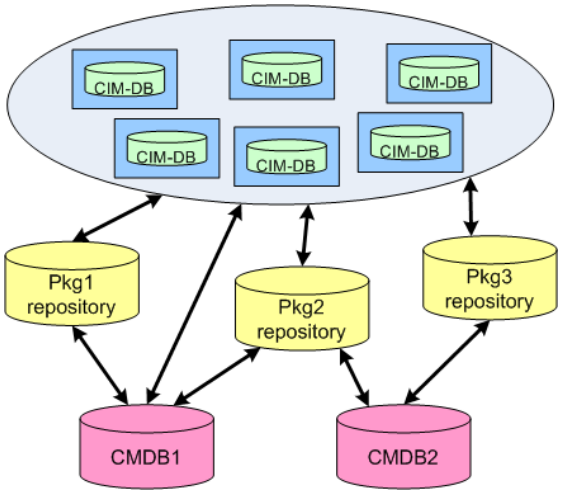

Fig. 2. Illustration of repository hierarchy for management

view of the configuration. They can be exploited for global health monitoring of the data center, and are beginning to be used for top-down control by interfacing with individual packages. For example, management of the network that includes routers from multiple vendors would require engaging the management capability built into each router. Interoperability between CMDBs between different vendors is a serious issue that some recent standards efforts such as CMDBf (see `dmtf.org/sites/default/files/standards/documents/DSP0252_1.0.1_0.pdf`) are attempting to address. Nevertheless, working with multiple CMDBs by different vendors provides ample opportunities for misconfiguration and attacks that can exploit top-down control functionality to make the entire infrastructure unstable.

In general, management operations involve two distinct types of activities: (a) operations support and (b) resource control. The former refers to activities such as installation, configuration, patching, repair, etc. whereas control refers to dynamic resource allocation and scheduling. In servers, the control is typically handled by the "in-band" side (i.e., by OS and middleware) that runs on the main processor, whereas the operations support is handled by the out-of-band (OOB) side, which runs on a separate management processor, traditionally known as basboard management controller (BMC). Although BMC was originally responsible for very basic functions such as temperature monitoring and fan control, it has lately evolved to perform quite sophisticated management functions [17]. A similar configuration is used in routers and switches where the control side handles route updates and the management side handles interface configuration, QoS setup, etc. The control and management sides increasingly require coordination for sophisticated management thereby making the OOB side vulnerable to attacks coming from the in-band side, hence making the configu-

ration management security more critical. VM management is a perfect example of fusion of operations support and control. While VMs are dependent on the hypervisor that runs on the main processor, it is desirable to integrate VM management with the traditional OOB management software so that functionality like power/temperature driven VM migration can be accomplished.

3 Security of Configuration Management Data

In this section we show why the security of CM data is critical and what unique security issues arise relative to such data. We also discuss several potential attack scenarios that hackers may attempt. Although attacks on CM data have not been common in the past, much of this has to do with relative obscurity and inaccessibility of OOB based management. As the standards continue to evolve towards a truly integrated and comprehensive management CM data will increasingly become a rich target of security attacks.

3.1 Attacks on Computing Infrastructure

In this section we highlight generic attacks on or misconfigurations of repositories at various levels of the computing infrastructure. A later subsection discusses attacks more specifically crafted toward disrupting the network per se. Because of the criticality of the CIM data, low level software using it (e.g., preboot utilities, OS drivers, etc.) may check for consistency and sanity of CIM data before using it. However, the action taken in case of problems is usually not very sophisticated and may either result in panic and restart, or require management console input to correct the value and continue. A restart can clear up bad data only for static and discoverable information. Depending on the operator to fix the problem could be unreliable and even dangerous. In many cases, proper functioning requires consistency across multiple entities, and deficiencies in checks could be exploited by attackers to create undetectable corruptions. Furthermore, since different configuration parameters are used at different times (e.g., service startup time or reboot time), some corruptions may not be discovered for a long time, and when they are, it may be very difficult to track down their root cause.

Data centers typically employ large number of assets of identical type, which means that if an attacker finds a single vulnerability, he/she can use it for a large scale attack. The increasing size of data centers and the practice of using only a few unique configurations to ease management burden makes this problem easier to exploit by hackers.

Higher level repositories such as those maintained by a management package may duplicate the information contained in individual server CIM repositories to varying degrees. Such duplication is desirable from an efficiency perspective since access to firmware managed information over web-services could be extremely slow. Furthermore, allocation or migration of an application requires a global view of the entire data center. Since data copy stored in higher level repositories is unlikely to be refreshed frequently from individual repositories, a hacker can

corrupt it with the assurance that the corrupted data will not be overridden for some time. Such an attack can cause the provisioning process to misbehave and result in traffic congestion, server overload, or other problems.

The most insidious attack on CM data is on the aggregate parameters maintained in the package repositories. Aggregate parameters often include simple counts (e.g., number of servers or VMs available for job scheduling) and averages/sums over multiple elements (e.g., average CPU utilization over all servers in a rack or total uplink traffic in and out of the rack). These aggregate parameters are useful for making quick decisions before examining the more detailed individual data. For example, if a provisioning request comes to a rack manager, it can be promptly rejected if its uplink BW requirements exceed the available uplink BW, or the average CPU utilization is too high. A corruption of such aggregate data can cause either underutilization of resources (e.g., no allocations take place from the rack because the used uplink BW is artificially set too high) or overutilization (e.g., congestion in the rack switch because the available uplink BW is artificially set too low). The attacker could even set the parameters to values that results in error exit or emergency shutdown (e.g., thermal limit exceeded) and thereby bring the entire rack down.

The risks posed by corruption of aggregate data become more severe as we go up the hierarchy. In particular, a corruption of a single configuration item in the global CMDB (e.g., total available bandwidth at top level switch) could bring down the entire data center. Unfortunately, the complexity of CMDB's presents a large number of such opportunities for severely disrupting the data center operations.

3.2 Attack Pathways

The most direct way of corrupting key configuration data is compromise of the servers running higher level management entities such as a cluster or rack manager. However, there are also several, presumably easier, ways of causing disruption. For example, a hacker could make legitimate requests that trigger known weaknesses of the configuration management procedures or by hiding malicious code into the requests. The latter is facilitated by the expanding use of web-services based management, which is known to be vulnerable to a host of attacks [30]. As with other web-services, WSMAN interfaces for manipulating configuration data are described using WSDL (Web Services Definition Language) and the descriptions are often automatically generated. Automatic generation often means that descriptions include *all* supported procedures including those not really intended for use by non-developers. Hackers can exploit these as an easy mechanism to corrupt configuration data. Similarly, publication of the web services via a public directory such as UDDI simplifies the job of the hacker.

Since SOAP headers in WSMAN commands use XML and require XML parsing, it is possible to craft bogus headers that nevertheless require significant parsing effort. Also, a SAX (Simple API for XML) based parser (that extracts all relevant information in a single pass) can be easily tricked into overwriting

earlier values. This is one mechanism by which a legitimate update to configuration variable can be hijacked to produce an invalid or otherwise problematic value. Another mechanism concerns the misuse of *XML external entities*, which is merely a macro facility by which one could include contents of external files in the XML stream. If the hacker can overwrite or replace such a file, it can put arbitrary XML code there. One such possibility is to open a new TCP connection with the privileges of the XML parser and perform arbitrary data transfer. A related attack is XML schema poisoning to alter control flow or otherwise cause incorrect processing of XML data. Finally, the hacker can inject arbitrary data wrapped in XML (e.g., XPATH expressions, SQL queries, LDAP requests, etc.) to achieve specific attacks related to how the configuration data is manipulated. Some of the XML manipulation attacks can even disable authentication and thereby gain unrestricted access to the configuration data.

3.3 Attacks on Network Configuration

As the management interface to switches becomes more directly engaged in supporting features such as QoS controlled virtual paths or entire virtual clusters, the corresponding configuration data becomes more vulnerable to attacks and misconfigurations. Switch configuration data includes a number of attributes including VLAN setup, size of address table and number of MAC addresses per port, QoS setup, interface speeds and corresponding parameters, etc. In particular, VLANs are often used for traffic isolation, and are being exploited to provide layer-2 QoS in the data center Ethernet context [5]. A number of VLAN related and other attacks on switches are well known and can be exploited to disrupt data center traffic flows [27]:

1. MAC flooding attacks that fill-up the hardware table that associates destination MAC addresses with switch ports. Once that happens, traffic directed to additional MAC addresses will be broadcast to all ports and can be easily sniffed. This issue is normally addressed via a configuration parameter that limits the number of unique MAC addresses per port. A corruption of this value can circumvent the protection.
2. Switch configuration anomalies or corruption can cause certain ports to start behaving like "trunk ports" that are allowed to forward traffic between VLANs.
3. Ethernet switches use some version of the spanning tree protocol (STP) to implement loop-free layer-2 routing. It is possible to corrupt the configuration data such that the bridge PDU (BPDU) messages used in the STP cause a switch to be elected as the root of the spanning tree and thereby have the traffic directed as desired.

With increasing use of Ethernet for storage traffic, these attacks (and several others not mentioned here) can even be used for large scale exposure or stealing of stored data.

Although data centers mostly employ layer-2 switches in the network interconnection infrastructure, layer-3 routers are also needed at the periphery to

connect to the external world. In addition, the emerging trend of distributed data centers to support seamless cloud computing environments also results in non-local server-to-server traffic flowing through the routers. In addition to the layer-2 attacks discussed above, routers can also be attacked at layer-3. The extensive configuration and policy setup for the interdomain routing algorithm such as BGP, coupled with the relunctance on part of ISPs to share their setups leads to plenty of chances of misconfigurations [21, 2]. Some common ailments include advertisement of route for an entire prefix which is not entirely served by the router, improperly configured alternate routes leading to packet loss and convergence issues, and incorrect packet discrimination rules that may deny desired packets or accept unwanted ones. These issues are often addressed by checking route configurations against the rules and policies expressed declaratively. Some recent research in area concerns discovering the rules and policies by data mining instead of being pre-specified [21].

Routers invariably support flexible IP flow and QoS configuration including MPLS, differentiated service (DSCP), reservations (e.g., RSVP), etc. The extensive configuration involved in this setup needs to be protected as well. There are many attacks that can be directed towards MPLS signaling [24], differentiated services [32], and reservation based QoS services [33] and mechanisms based on cryptographic authentication and encryption have been proposed. In all cases, however, the routers involved must store the necessary configuration parameters for the QoS treatment, and integrity of this data needs to be ensured.

The most common routing attacks concern perturbation of the routing table by latching on to route update messages. In particular, suppression, duplication, or change to route update messages can cause misdelivery and congestion. Such attacks have been considered extensively in the literature, and are not strictly speaking configuration attacks. Cryptographically enhanced route update protocols, such as Secure BGP (SBGP) [6]) are designed to secure updates, but require much more complex configuration. In fact, SBGP requires two PKI (public key encryption) hierarchies and is thus even more complex than DNS-SEC that requires single such hierarchy. It is to be noted that although DNS-SEC was introduced to address the vulnerabilities of DNS, its complexity makes it quite prone to misconfigurations [9].

Energy Related Attacks As more powerful devices and servers are stuffed in smaller form factors, the power and thermal densities become unsustainable and effective cooling more expensive. At the same time, the increasing size of data centers makes the energy/cooling and power delivery costs dominant in large data centers. This has resulted in aggressive push towards not only energy efficient design but also active management of energy/power as a finite resource that can be shifted where needed most. Thus, configuration management includes intelligent energy management and involves keeping track of energy availability, current requirements, cooling requirements, temperature, energy states of various devices and servers, etc [19]. This provides ample new opportunities for energy related attacks, but to date very little attention has been paid towards

the security aspects of power management. For example, in windows OS, although the power management can be restricted by administrators, by default any authenticated user can alter power management settings.

There are several ways to abuse power management capabilities. If the ACPI (advanced configuration and power interface) access is not tightly controlled, it may be possible for a hacker to gain access to a user account and set power states of the server to undesirable values (e.g., put an active server to sleep, run the CPU and other devices at lowest speed thereby causing congestion, prevent CPU/devices from going into lower power modes and thereby force thermal events, etc.) Another possibility is to exploit web-service security holes to corrupt the recorded values of energy states of servers in CMDBs and/or energy states of various devices (e.g., CPU, memory, IO adapters, etc.) CIM repositories. Finally, it is possible to cause power circuits to be overloaded or thermal emergencies to be triggerred by simply increasing the load – that is, by a form of denial of service (DoS) attack focussed on energy consumption. It is important to note that more sustainable and energy efficient designs make data centers more vulnerable to energy attacks [18]. For example, ambient cooling, use of lower capacity power supplies, lower capacity UPS, etc. all make the energy attacks easier.

3.4 Attacks on virtualized Device Configuration

As virtualization becomes pervasive in data centers, it brings new security challenges as pointed out in [26, 14]. These challenges imply similar issues with respect to the configuration data that defines the characteristics of the virtual device. While the number of real devices is limited by monetary, space, power and other considerations, virtual devices can be created at almost no cost. Thus the number of the virtual devices within a data center can grow explosively, particularly as the virtualization overhead is driven to be negligible via proper hardware support and software mechanisms [4]. Consequently, management servers have to handle many more devices which increases the amount of configuration data and is corresponding vulnerability.

Unlike physical devices, virtual devices may appear and disappear dynamically. This makes the tracking of compromised devices quite difficult since the device may disappear before an attack is discerned. Moreover, since the states and configurations of virtual devices can be easily logged and restored later, old configurations may result in inconsistencies and conflicts. Since the virtual devices run atop physical ones, they must bind both with the underlying physical infrastructure and the management servers. When a virtual device moves from one physical system to another, these binding relations change as well, and must be properly updated. A compromised virtual device could easily "infect" the configuration of its next host and possibly even the next local management server before the compromise is detected.

As the virtualization support permeates the hardware and firmware, virtual device configuration data is beginning to migrate to the standard management repositories, as already stated and protection of these repositories becomes essential. Although, isolation is one of the most important reasons for the popularity

of VM technology, it is possible for rogue processes running in a VM to escape the VM boundaries and compromise other VMs or the physical machine. Thus, it is important to protect VM configuration data against such "insider" attacks as well. The trusted virtual data center proposal from IBM [3] is designed to enforce isolation between VMs on a platform by providing a special management VM that checks access to specified resources (e.g., a virtual disk, VLAN, etc.) by a VM. The management VM interfaces with the configuration manager CIMOM and thus is able to protect the integrity of configuration data. It also proposes to virtualize the platform TPM (trusted platform module) in order to provide attestation capabilities to each VM for the software running on the VM. This work exploits the sHype mandatory access control extensions to the Xen hypervisor [28]. Reference [8] discusses how to establish trusted virtual domains for communication among a set of related VMs using existing techniques such as Ethernet encapsulation, VLAN tagging and VPNs.

4 Securing Configuration Management Data

In this section, we first briefly discuss available security mechanisms in current management standards and then introduce a novel scheme that exploits the redundancy that exists in the configuration data in order to detect updates that may result in misconfigurations or corruptions.

4.1 Existing CM Security Mechanisms

WBEM provides several mechanisms for secure access to CIM repositories. Clients can be authenticated via authentication tokens which are usually user-name/password, but could also be Kerberos tokens or public key (PKI) certificates. Message integrity is provided by including a message digest in the communication linked with the authentication token. ACLs are used for providing required access control to the clients over CIM hierarchies and must be configured manually. By default, all clients have read access over the entire name space. The auditing functionality records all authentication events (successful or not) and changes to management data made by the clients.

WS-security [34] (a part of WBEM) supports authentication, integrity and confidentiality for SOAP messages via a variety of security models including PKI, Kerberos, and SSL. Specifically, it supports multiple security token formats, multiple trust domains, multiple signature formats, and multiple encryption technologies. An extension to WS-security [35] supports secure message exchange between multiple parties by establishing a security context for the entire message exchange session.

Authentication of clients and devices via PKI certificates is clearly useful but its high overhead usually detracts from its routine use. For the most part, configuration data does not need to be kept confidential; instead, the primary requirement is its integrity. Therefore, instead of encrypting the stored or exchanged data, it is more important to address the issue of data corruption –

whether intentional or accidental. Furthermore, when the corruption does happen, it should be possible to detect and possibly correct it. Suitable schemes depend on the nature of data in terms of how often it changes and the extent of damage that corruption can cause.

4.2 Exploiting Redundancy in Data

The configuration data stored in various repositories has significant amount of redundancy which can be exploited for consistency verification purposes. Let us start with the relatively static information such as hardware type, setup and raw capacities (e.g., 3.0GHz, dual-core processor, 16GB DDR3-1333 memory, etc.), installed software (e.g., Redhat EL6), etc. Data centers often deploy a large number of identical and identically configured servers, switches, storage bricks and other assets. This lack of diversity can be exploited both by the hackers for replicating attacks and by the configuration manager for detecting configuration attacks and to reduce the overhead of protection. In general, the configuration of all servers (or devices) of the same type may not be identical, instead, certain aspects of the configuration may be unique (e.g., switch port to which the server is connected) while others are either identical or admit some narrow range or set of acceptable values. We call these parts as unique configuration data (UCD) and shared configuration data (SCD) respectively. Although both types of configuration data need to be protected, it is possible to exploit the existing redundancy for protecting SCD at a lower cost. This could be done by nominating a few assets as *reference* whose SCD instance is regarded as a gold standard and used for validating others. The configuration data of the reference assets is protected aggressively using the following mechanisms:

1. Authentication of all HW and SW installed on the assets via local TPM (trusted platform module), if available, or via suitable remote authentication.
2. Extensive sanity checking of repository data both against actual configuration (obtained via discovery and enumeration procedures) and via consistency conditions.
3. Information flow related security controls such those suggested in [20] to monitor attacks on web-services that can catch and even recover from undesirable flows.

Certainly the UCD part of the data cannot be left unprotected, but to use the same mechanism, it would have to be duplicated in the reference asset. This is a reasonable approach if the amount of unique data is very small. If the unique data can be verified more directly via a direct discovery or enumeration, than such duplication is not necessary.

The configuration data from assets can be checked against that of the reference assets both periodically and at the time of significant events (e.g., just before and after a reboot, or when a potential attack/compromise is suspected). Hashing can be used to implement this efficiently in trusted management servers; i.e., the management servers only store a hash value (or more generally a hash

map) of the data in each repository, and a comparison of hash maps is used to check for consistency. The hashing scheme can also admit ranges if the values are trimmed down to the lower bound of the range before computing the hash. This is an acceptable approach if the allowed ranges are rather narrow and the precise value within the range is inconsequential from a configuration robustness perspective. For wider ranges, it may be possible to divide the assets into groups and maintain a separate reference for each group.

A similar scheme can be used for virtual machines as well. Although it is possible to create VMs with arbitrary parameter values or even modify some parameters for a running VM, such an approach quickly leads to a management nightmare in large systems. Thus, using only a rather limited candidate set of VM configurations is not only a practical approach, it also allows the use of above mentioned group based protection mechanism.

When multiple groups are involved, an update to a changeable configuration parameter may require moving the entity to a different group. This can be allowed safely provided that the group membership is maintained in the secured management server that implements the authentication and verification, rather than residing with the entity itself.

4.3 Securing Aggregate Data

As stated earlier, the security of aggregate data is vital. Fortunately, aggregate data is by definition redundant and can be verified via recomputation either periodically or when the values fall outside some nominal range. If the underlying algorithms update the aggregate data lazily (i.e., not every change to individual values is immediately reflected in the aggregate values), one could not insist on exact match on verification. This could lead to false positives, whose probability can be reduced by the following measures: (a) Use of multiple verification failures to declare an attack while each time still replacing the existing value with the newly computed value, and (b) Use of historical information or "trend" to distinguish normal behavior from abnormal one. There is obviously a tradeoff between false positive probability and latency in detecting an attack.

It is important to note that attack resistance requires a somewhat different treatment of error conditions with respect to aggregate quantities: "impossible" or out of range values should not result in simply raising an exception; instead, a recovery attempt needs to be made by recomputing the aggregate value.

5 Challenges in Securing Configuration Management

In this section we discuss a number of issues that complicate configuration management and describe the security challenges brought about by them.

5.1 Configuration Dependencies

The configuration and hence proper operation of an entity could depend on the configuration of another entity either directly or due to the way system is struc-

tured. In some cases, a common configuration instance may apply to multiple entities. The classical case for this is configuration specified via configuration files (e.g., the current situation with VMs). In this case, a misconfiguration can affect many seemingly independent enties. A similar situation arises whenever a common default configuration is used, even when each entity has its own independent configuration record. The problem here is that many instances will likely to continue to use the default configuration that may be faulty. In both of these examples, it may be possible for a hacker to deliberately cause misconfiguration by altering the common information. The practice of keeping aggregate data also creates a strong dependency in that the corruption of the aggregate value affects proper usage of the corresponding assets. Dependencies can also be indirect – for example, misconfigured network BW between two racks may prevent allocation of certain applications to these racks even if the individual servers in the racks have adequate computing and network capacity. More generally, misconfiguration of a single device (e.g., a NIC, accelerator, etc.) could substantially impact all applications that use that device or use it in a certain way.

A major challenge in configuration management is characterization and modeling of such dependencies. Although there is a substantial amount of literature on identifying program data and control dependencies [31], the dependencies from the misconfiguration impact perspective can be different, perhaps more in line with the notion of abstract dependencies that is often used in the context of program slicing [23]. New metrics and models to characterize such dependencies and to identify "crtical dependencies" are the first steps to find out ways of fortifying the CM systems and enhancing the robustness of CM data. The notion of "critical dependencies" can be defined in multiple ways including dependencies that lead to most wide-spread impacts, most serious impact, or even those that are easiest to exploit for disrupting system functioning. A formal characterization and algorithms for finding critical dependencies remain important open problems.

A somewhat unique aspect of dependencies in configuration data is that many dependencies may not be triggered for long periods of time. For example, updates to the physical device configuration within a server will, in most cases, not take effect until the server is rebooted. In a data center environment, servers may not be rebooted for weeks or months. This means that by the time the misconfiguration or malicious update takes effect, it may be nearly impossible to track down how, when, and under what circumstances the update happened and what else might have been affected. In terms of VM configuration, certain parameters may be fixed for the duration of VM's existence (e.g., size of allocated hardware threads or main memory) while others are more dynamic (e.g., allocated disk space) depending on the design. At the application and middleware level also, different parameters may have different lifetimes. Thus, in order to properly model the dependencies, we also need to consider grouping by events that trigger them, while at the same time characterising the relationship between various groups.

5.2 Configuration Management for Clouds

Configuration management in a cloud computing environment brings in additional challenges due to involvement of multiple parties and possibly distributed infrastructure. Although current cloud computing infrastructures are owned and operated by a single vendor (e.g., Amazon), a more general outsourced model is likely to emerge in the future as discussed in [19]. For example, several different enterprises may lease "data centers" as physical clusters from the same underlying server farm. The advantage of such a model is the economy of scale for the server-farm operator, and the ability of the leasing enterprise to easily alter the size of its data center and delegate physical infrastructure management (e.g., replacement of failed servers or patching them) to the server farm operator. These outsourced physical data centers could then be virtualized and leased to the service providers or end customers. In this model, it is possible for the same service provider or customer to lease multiple, possibly gegraphically distributed, resources and thereby create a distributed, virtualized data center. Fig. 3 shows such a 4-layer conceptual model of future data center. In this depiction, rectangles refer to software layers and ellipses refer to the resulting abstractions.

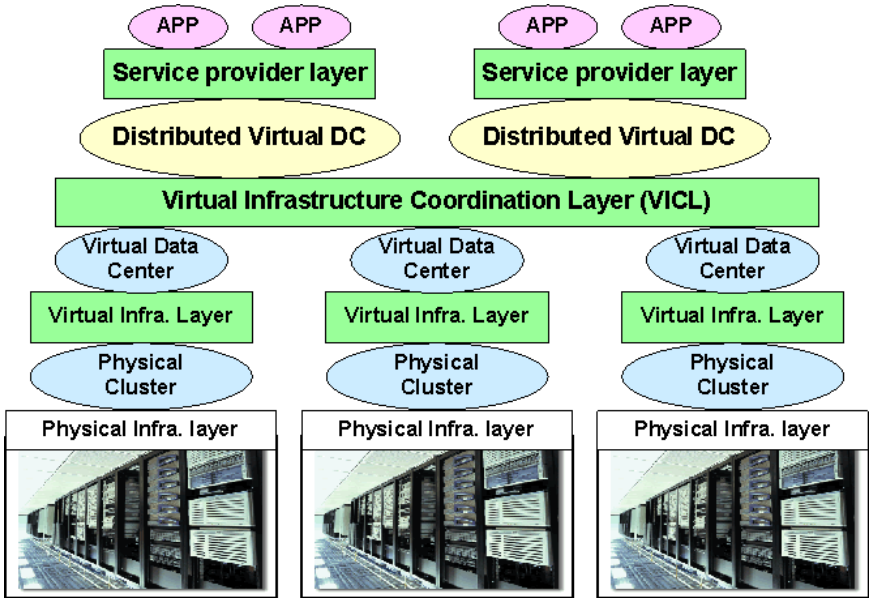

Fig. 3. Logical Organization of Future Data Centers

Such a model subsumes everything from a non-virtualized, centralized data center entirely owned by a single organization all the way up to a geographically distributed, fully virtualized data center where each layer possibly has a

separate owner. The latter extreme provides a number of advantages in terms of consolidation, agility, and flexibility, but it also poses challenges in a number of areas including the integrity and privacy of the configuration management. In particular, an effective resource and configuration management requires that nonlocal CM data be available whereever sharing happens. This implies, for example, that the physical clusters need visibility into the configuration of the networking infrastructure that happens to be shared (horizontal visibility). Similarly, a virtualized data center running on top of a physical cluster may need to know both about the configuration of the underlying physical cluster (vertical visibility) and other virtual data centers (horizontal visibility). However, such visibility needs conflict with the privacy and security needs of various layers. The latter may include the following:

1. Visibility restriction to Higher Layers: A physical cluster owner may not want to fully release configuration information to the virtual data center above, because part of this information may be considered as a "business secret".
2. Lack of Trust in Lower Layer Information: A virtual data center may not entirely trust what is being revealed by the lower layer. Generally, there is no expectation of malicious intent by the lower layer, but the lower layer may underreport or hide certain resources.
3. Lack of Trust in Sharing Information with Lower Layer: The higher layer may not give out all of its configuration data to the lower layer, because such data may be passed on to its peer by the lower layer.
4. Complete lack of knowledge or trust horizontally and hence no direct information sharing among the peers.

Achieving effective configuration management while respecting the privacy/security needs of various layers is a challenging problem that has not been examined in the past. It is important to note here that since arbitrary parties may be sharing the infrastructure, there is a far greater chance (than in a traditional data center environment) of inadvertent or intentional misconfigurations that affect other parties in terms of performance and even functionality. Of course, such an environment also provides ample opportunities for customers to actively launch security attacks. Note that while more information sharing allows better allocation decisions, it also increases risk of uncooperative behavior.

As a simple example, suppose that two virtual data centers, say, V_1 and V_2, are deployed on a physical cluster P consisting of two racks. If there is a strict control over all resources, and the resource limits for V_1 and V_2 are conservatively specified, then most of the undesirable interactions can be avoided. However, this is very difficult or infeasible in practice. For example, it is neither possible, nor desirable to provide BW controlled virtual path between the two racks for V_1 and V_2 separately. The same applies to how much power draw or heat dissipation V_1 and V_2 are allowed or how much memory bandwidth they can use. In the absence of tight resource controls, if the physical resource allocation information is not shared among V_1 and V_2, each of them could inadvertently allocate their

tasks so as to stress the inter-rack network BW. On the other hand, if the allocations are shared, it is possible that V_1 (or V_2) intentionally squeeze inter-rack communications of the other either to gain performance for their applications or to hurt others. Note that the reason why this is a unique problem for clouds is because the task allocation decisions are no longer done by a single central authority.

In addition to the problem of multiple virtualized data centers sharing common resources, the model in Fig. 3 also admits truly large, geographically distributed data centers (virtualized or not). Configuration management in such an environment becomes challenging because of potential lack of unformity among the configuration management structures and practices used by different server farms, which further increases chances of misconfigurations. Reference [13] describes a distributed configuration management approach for routers in a large ISP network using a specifically designed template based configuration management language. There are proposals to simplify management of large scale systems by partitioning the management responsibilities either physically (as in[25]) or logically (as in [1] where data plane presents a standard management interface to the management plane). However, in general, scalable and secure configuration management in large heterogeneous distributed environments remains an open problem.

5.3 Hardening Configurations Against Attacks

Configuration management can be made more robust both by reducing the attack surface and by changing it dynamically. The attack surface can be reduced by making the configuration data less accessible via a variety of techniques including authentication, encryption, access controls, and information flow controls, as already discussed. The change of attack surface can be done by exploiting moving target defence (MTD) or more generally by introducing diversity. In particular, the scheme discussed in section 4.2 can be hardened by rotating the reference nodes according to a randomized scheme, so that brute force attacks on the reference server have less chance of success.

The most generic diversity technique is to change in-band and OOB IP addresses and port numbers for each machine in the management hierarchy according to some schedule. However, there are other configuration management specific ways of using MTD as well. One potentially weak link in CM is the interaction between OOB and IB sides which is usually via the IPMI (intelligent peripheral management interface). Hardening of IPMI messages by techniques similar to instruction set randomization (ISR) and consistency checking could be useful in foiling attacks that target IB-OOB interaction.

It was commented in earlier sections that the dynamism brought in by virtualization (e.g., on-the-fly resizing of VM resources and migration) complicates configuration management and enhances opportunites for attacks and misconfiguration. However, the same dynamism can also be exploited to fortify the defenses. For example, a deliberate VM migration or changes to resources allocated to the VM can make it harder to target the VM. This also makes attempts

to poison VM configuration data harder. However, there is currently no formal model that provides a basis for choosing one technique over another. Similarly, quantification of the benefits of dynamism remains an open problem.

Randomization of memory addresses, or more generally address space layout randomization (ASLR) is a well known defensive technique that has also been implemented in some currect operating systems. The same principle can be used for CIM objects representing configuration. The base offset of CIM objects can be randomized relatively easily, but a finer grain randomization of individual attributes within a CIM object is harder. Once again, quantitative models for guiding design of good randomization techniques and techniques for assessing their effectiveness remain largely unexplored.

It was mentioned earlier that configuration errors/corruption can be particularly insidious since configuration data is often used on special events (e.g., reboot of machine or restart of service). A more proactive use of configuration data can detect problems early, but needs to be applied carefully to avoid inconsistencies and early failures. For example, a systematic approach to testing out altered configurations in a continous but limited way could be a useful way to provide *timing diversity* that spreads the risk across time.

MTD can be potentially useful in resolving the conflict between usefulness and risks of sharing configuration data horizontally or veritcally in the context of the general cloud computing model discussed in section 5.2. In particular, the nonlocal configuration data can be fuzzified before sharing so that various entities can make good allocation decisions without having to share their detailed configuration data. This fuzzification cannot be a simple aggregation; it needs to change dynamically so that the information less useful for abuse. Once again, coming up good mechanisms for fuzzification and how to achieve balance between usefulness and privacy remain open problems.

6 Conclusions

In this chapter, we examined the problem of securing configuration management (CM) within data centers. We discussed CM related vulnerabilities in a data center in general, including servers and network nodes. It was noted that the web-services based management – although increasingly popular – can harbor attacks that can seriously disrupt data center operations. We then presented some simple mechanisms to ensure integrity of CM data. However, the general problem of CM security remains unsolved, particularly in the emerging cloud computing environment. We presented the challenges of securing CM and pointed out several approaches that use the principle of moving target defense.

References

1. H. Ballani and P. Francis, "CONMan: taking the complexity out of network management", Proc. of ACM SIGCOMM Workshop on Internet Network Management, Sept 2006, pp41-46

2. L. Bauer, S. Garriss, M.K. Reiter, "Detecting and resolving policy misconfigurations in access-control systems", In Proc. of 13th ACM Symposium on Access Control Models and Technologies, June 2008, pp185-194.
3. S. Berger, R. Cceres, D. Pendarakis, et al., "TVDc: managing security in the trusted virtual datacenter", SIGOPS Oper. Syst. Rev. 42, 1 (Jan. 2008), pp 40-47.
4. K. Biswas and A. Islam, "Hardware Virtualization Support In INTEL, AMD And IBM Power Processors", available at `arxiv.org/abs/0909.0099`.
5. IEEE task group 802.3.az, "Energy Efficienct Ethernet", `www.ieee802.org/3/az/public/nov07/hays_1_1107.pdf`.
6. K. Butler, T. Farley, T. McDaniel, J. Rexford, "A Survey of BGP Security Issues and Solutions", to appear in Proc. of IEEE, 2010.
7. "Common Information Model", Available at `www.wbemsolutions.com/tutorials/CIM/cim-specification.html`
8. S. Cabuk, C.I. Dalton, H. Ramasamy, M. Schunter, "Towards automated provisioning of secure virtualized networks", Proc. of 14th ACM CCS conference, Oct 2007, pp 235-245.
9. C. Doccio, J. Sedayao, K. Kant and P. Mohapatra, "Quantifying and Improving DNSSEC Availability", to appear in proc. of ICCCN conference, Aug 2011.
10. "Virtualization Management (VMAN) Initiative : DMTF Standards for Virtualization Management", Available at `http://www.dmtf.org/standards/vman`
11. "Open Virtualization Format", Available at `dmtf.org/sites/default/files/standards/documents/DSP2021_1.0.0.tar`
12. J. Crandall, "DMTF Technologies Overview", Available at `www.snia.org/events/storage-developer2008/presentations/wednesday/JohnCrandall_DMTF_Profiles_for_Storage.pdf`
13. W. Enk, T. Moyer, P. McDaniel, et.al., "Configuration management at massive scale: system design and experience", IEEE Journal of Selected Areas in Communications, April 2009, Vol 27, No 3, pp323-335.
14. Tal Garfinkel and Mendel Rosenblum, "When Virtual Is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments", USENIX Association, 2005
15. P. Goyal, R. Mikkilineni, M. Ganti, "FCAPS in the business services fabric management", Proc. of 18th IEEE Intl. workshop on Enabling Technologies, 2009.
16. R.C. Merkle, "Protocols for Public Key Cryptosystems", In Proc. of 1980 IEEE Symposium on Security and Privacy, 1980.
17. Intel Active Management Technology. Available at `en.wikipedia.org/wiki/Intel_Active_Management_Technology`
18. K. Kant, "Distributed Energy Adaptive Computing", Proc. of International Conf. on Communications (ICC), May 2010.
19. K. Kant, "Data Center Evolution: A Tutorial on State of the Art, Issues, and Challenges", Elsevier Computer Networks Journal, Dec 2009.
20. M.S. Lam, M. Martin, B. Livshits, J. Whaley, "Securing Web Applications with Static and Dynamic Information Flow Tracking", Proc. of ACM sigplan symp. on partial evaluation and semantics based program manipulation (PEPM), 2008.
21. F. Le, S. Lee, T. Wong, et. al, "Detecting network-wide and router-specific misconfigurations through data mining", IEEE/ACM Trans. on networking, vol 17, No 1, Feb 2009, pp 66-79.
22. C. E. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efcient Supercomputing", IEEE Trans. on Computers, Vol 34, No 10, pp892901, 1985.

23. I. Mastroeni and D. Zanardini, "Data Dependencies and program slicing: from syntax to abstract semantics", Proc. of ACM sigplan symp. on partial evaluation and semantics based program manipulation (PEPM), 2008.
24. F. Palmieri and U. Fiore, "Enhanced security strategies for MPLS signaling", Journal of Networks, Vol 2, No. 5, Sept 2007.
25. L. Pasquale, J. Laredo, H. Ludwig, et.al., "Distributed Cross-Domain Configuration Management", Proc of ICSOC 2009, LNCS 5900, pp622-636.
26. J.S. Reuben. A Survey on Virtual Machine Security. Helsinki University of Technology, 2007. Available at `http://www.tml.tkk.fi/Publications/C/25/chapters/Reuben_final.pdf`
27. S.A. Rouiller, "Virtual LAN security: weaknesses and countermeasures", available at `uploads.askapache.com/2006/12/vlan-security-3.pdf`
28. R. Sailer, T. Jaeger, E. Valdez, et al, "Building a MAC-based Security Architecture for the Xen Opensource Hypervisor", 21st Annual Computer Security Applications Conference (ACSAC), Dec 2005.
29. F.T. Sheldon and C. Vishik, "Moving toward trustworthy systems: R&D Essentials", IEEE Computer magazine, Sept 2010, pp 31-40.
30. A. Stamos and S. Stender, "Attacking Web Services: The Next Generation of Vulnerable Enterprise Applications", Proc. of Defcon XIII. Available at `www.isecpartners.com/.../iSEC-Attacking-Web-Services.DefCon.pdf`.
31. W. Stanley, J. Laski, "Program Dependencies", in Software Verification and Analysis, springer-verlag, 2009, pp125-142.
32. A. Striegel, "Security Issues in a Differentiated Services Internet", Proc. of HiPC workshop, 2002.
33. V. Talwar, K. Nahrstedt, S.K. Nath, "RSVP-SQOS : A SECURE RSVP PROTOCOL," Proc. of IEEE Intl. conf. on Multimedia and Expo (ICME'01), 2001
34. Web service security specification, available at `docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf`
35. Web services secure conversation specification, available at `specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf`