Security Considerations in Data Center Configuration Management

Krishna Kant, Meixing Le and Sushil Jajodia
George Mason University
Fairfax, VA, 22030
{kkant, mlep, jajodia}@gmu.edu

***Abstract*—Data centers need to manage a large amount of configuration information for a variety of computational, storage and networking assets at multiple levels (e.g., individual devices to entire data center). The increasingly sophisticated configuration management required to support virtualization significantly enhances chances of misconfigurations and exploitation by hackers that could impact data center operations. In this paper, we expose a number of attack/misconfiguration scenarios for data center resources. We also propose a mechanism, called Sentry, for securing this data by exploiting the hierarchical setup and redundancy in the server and network configurations. We show that the scheme can secure configuration data with only a small overhead.**

Keywords: Configuration Management; Security; Data Center; Common Information Model

I. 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, etc. Each of these assets must be managed at levels spanning from individual devices and device drivers all the way up to the entire data center infrastructure and operations. The configuration data is often required from multiple perspectives including performance, availability, security, energy efficiency, etc. and over the entire *life cycle* of the hardware and software, starting from their initial deployment to the ultimate retirement. This is the scope of what is loosely referred to as "configuration management" (CM) and it involves multiple repositories to hold a variety of configuration information that must be protected from unauthorized access and corruption. Traditionally, CM repositories have been relatively isolated and available only to the physically separate management infrastructure; however, the isolation between management and normal in-band resource management is swiftly breaking down due to virtualization and dynamic resource management.

In this paper, we expose CM vulnerabilities and discuss mechanisms to make it more secure. We also exploit the special structure of data centers in order to provide a low overhead mechanism for protecting the configuration of servers and the data center network. To the best of our knowledge, this is the first study of its kind for securing the data center configuration repositories that may reside both on out-of-band and in-band sides.

The outline of the paper is as follows. Section II discusses the configuration management infrastructure in data centers and its pain points. Section III discusses in detail the consequences of attacks on CM data and the mechanisms that hackers can exploit for attacking it. Section IV discusses some general security mechanisms for protecting CM data, and introduces a novel mechanism called Sentry for securing the data center assets. Section V applies this mechanism to secure data center assets and quantifies its overhead and performance. Section VI discusses the related work. Finally, section VII concludes the discussion.

II. Data Center Configuration Management

Modern data centers invariably have the hierarchical structure that may include individual assets, chassis, rack, and clusters of racks, and this structure directs the networking and management infrastructure as well. Data centers also include a significant level of redundancy in order to provide high availability. This redundancy could be either explicitly configured (e.g., duplexed switches and uplinks for storage and regular network traffic) or available by virtue of ability to migrate workload from one resource to another.

Successfully operating a large data center involves large amounts of configuration data relating to servers and their components, network, storage, system software, applications, services, etc. In addition, auxiliary data is typically maintained to ensure smooth operations including data relating to performance, faults and failures. A systematic management of this data and its protection is crucial for the operations. The current data center practices in this area derive from well-known practices in the telecommunications systems. The latter follow the FCAPS model specified in ITUs telecommunications network management (TMN) standards [8] which enumerates five categories of management activities: Fault, Configuration, Availability, Performance, and Security. In the current data center context, we also need to add energy management and hence speak of "FCAPSE" model.

Although "configuration" has a specific meaning in the FCAPSE context, the widespread use of virtualization and dynamic resource management makes "configuration" a central piece for all aspects of managing the data center infrastructure and the services provided by it.

Configuration management involves a variety of repositories that generally follow the hierarchical system structure (e.g.,

device level, server/switch level, chassis level, etc.) The low level repositories are usually in firmware, but those above the level of individual assets (servers, switches, etc.) use regular databases. These repositories generally follow the standard Common Information Model (CIM) developed by distributed management task force (DMTF) [5]. CIM is a hierarchical modeling language based on UML (unified modeling language) and provides 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.

DMTF has also developed Web-Based Enterprise Management (WBEM) specification that provides mechanisms to exchange CIM information in an interoperable and efficient manner. The components of WBEM include representation of CIM information in a variety of ways including the popular web services based management (WSMAN) which runs atop SOAP (simple object access protocol). SOAP is itself typically is layered on top of HTTP. WSMAN provides capabilities of addressing, data communication, events interface, and security (authentication and encryption) services for SOAP communications. However, 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.

Although CIM based models can be used for representing the structure of entities beyond individual HW assets (e.g., for racks, clusters of racks, virtual clusters, installed software and services, etc.), the higher level entities require rich data management capabilities (e.g., operations based on data contained in multiple repositories). These are typically implemented by management packages, each specializing in a specific domain. 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. VSMPs are also typically accessed via web services and thus are vulnerable to web-service attacks. Furthermore, the duplication of information across databases can cause inconsistencies and misconfiguration.

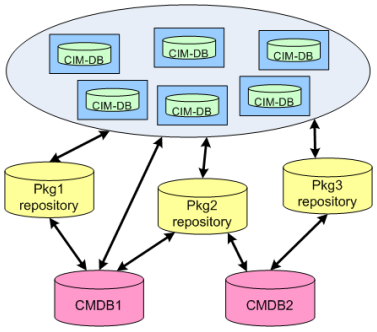

Fig. 1. Illustration of repository hierarchy for management

Most data centers also deploy one or more top level configuration management databases (CMDBs) that consolidate information from all repositories. CMDBs are important to capture dependencies between various packages and HW/SW components and provide a global view of the configuration required for health monitoring of the data center in cooperation with other repositories. Current CMDB's support federated model which can allow them to scale to large installations. Figure 1 shows the typical hierarchy of repositories in a data center. It shows two different CMDBs, perhaps from two different vendors to serve different purposes.

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 the usage management of resources. 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 operation support is handled by the out-of-band (OOB) side, which runs on a separate management processor. 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 configuration management security more critical.

III. 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 are not common today, much of this has to do with relative obscurity and inaccessibility of such data in traditional systems. However, the developments noted in section II suggest that CM data could soon become a rich target of security attacks.

A. Consequences of CM Attacks

Most security attacks target specific vulnerabilities associated with specification or implementation of protocols, OS services or applications (e.g., TCP SYN attacks or buffer overflow attacks) and thereby manage to disable or disrupt IT equipment that has those vulnerabilities. In contrast, an attack on CM data can have a global impact and disrupt operation of IT assets that remain otherwise untouched by the attack. In particular, simple attacks can bring an entire data center to its knees, and common error handling mechanism could make matters worse. In this subsection, we illustrate these hazards. The actual attack methods themselves are discussed in later subsections.

Let us start with attacks on or misconfiguration of CIM repositories within a server that corrupt CIM data. (Many of these apply to management and control side of switches and routers as well.) 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 problem 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 is clearly detrimental to operations and will usually clear up bad data only for static and discoverable information. Depending on the operator to fix the problem could be unreliable and even dangerous. Sometimes the attack on CM data may not be discovered until reinitialization or

rediscovery when the source of attack/misconfiguration is no longer traceable.

When the consistency checking is dependent on other devices, a verification may not even be possible. For example, if jumbo frames are enabled on a NIC but the switch on which the NIC homes either does not support jumbo frames or does not enable them, the verification is more complex and may not happen. Hackers can exploit such weaknesses in order to cause disruptions in the data center network.

Data centers typically employ large number of assets of identical type, which means that the same type of vulnerability exists on many assets. The increasing size of data centers makes this problem more serious and 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 such data is unlikely to be refreshed infrequently 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 active VMs or physical servers, number of used buffer slots, etc.) 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. A corruption of such aggregate data can ensure that either no allocations take place from the rack (used uplink BW set too high) or the rack switch is congested (used uplink BW set too low). The attacker could even set the parameter to a value that results in error exit and thereby bring the rack manager 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 data center operations.

B. Attacks on Switches and Routers

Switch configuration data includes a number of attributes including VLAN setup, size of address table and number of MAC addresses per port, L2 QoS setup (assuming data center Ethernet capabilities), interface speeds and corresponding parameters, etc. A number of attacks on switches have been identified, many of them having to do with VLANs [13], such as MAC flooding attacks, Spanning tree protocol (STP) based attacks, etc.

The above attacks (and several others not mentioned here) apply to routers as well; in addition, routers, in general, can have extensive layer-3 configuration data that can be attacked. Indeed, extensive literature exists on security attacks and potential solutions with respect to the routing protocols [4], flow labeling and QoS mechanisms [11], [16], [17], and misconfigurations [9], [2], [4]. However, since our focus is on data center network where routers play a rather limited role, we do not delve into the details of router protection.

C. Web Services Based Attacks

Web-services based management is becoming very popular due to the availability of a wide variety of software development tools. Unfortunately, this also makes configuration repositories vulnerable to a host of attacks on web services [15]. 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 disable authentication and thereby gain unrestricted access to the configuration data.

D. Attacks on virtualized Device Configuration

As virtual assets becomes pervasive in data center, they bring new security challenges as pointed out in [12], [7], and the number of the virtual assets within a data center can grow explosively. Unlike physical assets, virtual assets may appear and disappear dynamically. This makes the tracking of

compromised assets quite difficult since the asset may disappear before an attack is discerned. Moreover, since the states and configurations of virtual assets can be easily logged and restored later, old configurations may result in inconsistencies and conflicts. Moreover, when a virtual asset moves from one physical system to another, the binding relations between the virtual asset and the physical one must be properly updated. Furthermore, a compromised virtual asset could easily "infect" the configuration of its next host and possibly even the next local management server before the compromise is detected.

In addition, the configuration data of virtualized assets is usually maintain in normal files in stead of firmware CIM repositories. Besides being very inefficient to access and manage, file based configuration is much more vulnerable to attacks than the well managed CM repositories. If an attacker gains the privileges to the storage or file system, he can manipulate the configuration data freely.

IV. Securing Configuration Management Data

In this section, we survey the existing security mechanisms in WBEM and then present a novel scheme for protecting the configuration data in data centers.

A. 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. The auditing functionality records all authentication events (successful or not) and changes to management data made by the clients.

WS-security [18] (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 methods. An extension to WS-security [19] supports secure message exchange between multiple parties by establishing a security context for the entire message exchange session.

Authentication of clients and assets 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. Furthermore, an automated masking or "correction" of corrupted/tampered data is usually not necessary; instead, all we need is a quick and effective mechanism to detect such corruption/tampering. The mechanisms considered in this paper are motivated by these needs.

B. Protecting Data Center Configuration Data

A data center typically has many identical assets which may be grouped according to their use. When data centers add new equipment, they usually buy a group of assets from the same supplier with identical or similar specifications. However, assets bought at different times or from different suppliers may differ considerably. Therefore, one could typically expect multiple groups of assets, with assets within a group having same or similar configuration but those in different groups with quite different configuration. This lack of diversity can be exploited for enhancing robustness of configuration data at a much lower cost. The main idea is to select an asset as a reference node for each group whose configuration is regarded as a gold standard and used for validating others within the same group. The configuration data of such nodes is protected much more aggressively. Whereas, not all configuration parameters could be identical even within a group – some parameters such as the MAC or IP address must necessarily be unique to each asset. Thus, configuration parameters within a group can be divided into the following 3 categories:

1) Exact match parameters: these are the parameters that must remain identical for all the assets within the group. Any violation indicates a possible problem.
2) Range parameters: these parameters take values either from a small discrete set (e.g., NIC speed of 100 Mb/s, 1 Gb/s and 10 Gb/s) or a relatively narrow range (e.g., idle power). The parameters taking a small set of discrete values can be easily mapped to a small integer range (e.g., as an enumeration); therefore, we consider those also as "range" parameters. If such parameters go out of their valid range, that indicates a possible problem.
3) Unique parameters: these parameter are unique to each member of a group (e.g., IP address, etc.). We assume there is but a small number of these parameters and updates to them are infrequent. Unauthorized updates to these parameters could cause serious problems.

Although we motivated this categorization based on physical assets, it generally applies to virtual assets and software as well. For example, virtual machine parameters (e.g., amount of memory or number of hardware threads) usually follow one of a few predefined values, instead of randomly chosen values. For different type of parameters, we use different mechanisms to protect or check their integrity as discussed in the following.

The exact match parameters are protected via comparison of their hash values against that of a Gold standard, whereas unique parameters are protected by only allowing secure updates to them. The range parameters are protected by exploiting locality sensitive hashing (LSH) [21], that we explain next.

Given n range parameters for an asset within some group, this asset (point, node) can be represented as a vector in the n-dimensional space, denoted $V_r = [p_1, p_2 ... p_n]$ where p_i is the value of the ith parameter. If the group has K assets, we have K vectors which need to be clustered together. This is where LSH comes in handy. LSH functions are defined for a metric space, there are a threshold $r > 0$ and an approximation factor $c > 1$. A LSH function h is a hash function designed such that for two distinct vectors p and q, if the distance metric $d(p, q)$ is less than the threshold r, then $h(p) = h(q)$ with probability at least P_1 (where P_1 is usually close to 1). That

is, if the two vectors are "close enough", then their hash values will collide with a high probability. Also, if $d(p, q) \geq cr$, then $h(p) = h(q)$ with probability at most P_2 (where P_2 is usually close to 0). That is, if p and q are sufficiently different then their hashes collide only with a very low probability. Here the factor c essentially defines the transition region. A small transition region results in higher space and time complexity for the hash computation.

In our application of LSH, a group is predefined and we compute a group central point which is the average vector of all the vectors in the group. The problem is to find the smallest r value which we call the *radius* of the group so that all assets of the group are within the distance r from the group central point, and any point from other groups is further than r from the group central point. Since points with different dimensions will appear very far in distance in the LSH function and very different values on the same dimension will also makes the vectors looks far from each other, it is usually possible for the algorithm to find such r value. In the case that two group central points are too close to each other that the two groups have overlaps, it is suggested to merge the two groups together.

C. Sentry: A Mechanism for Securing Configuration Data

In this section, we present the Sentry, which is a mechanism to secure configuration data in data centers. The threat model of Sentry makes the assumption that the attackers can compromise the configuration data of the assets, but they cannot take full control of the assets. (The reason for this assumption is that detecting and isolating assets taken over by a malicious entity requires completely different mechanisms.) It is possible that the attackers exploit the aforementioned web services based attacks to compromise and forge the SOAP messages to misconfigure the assets.

In general, the management could involve a hierarchy of management servers (MS) such as the one shown in Fig. 2. Here we take the example of switches in a fat-tree [10] structure, but a similar approach can be used in other cases as well. In Fig. 2, we show a global management server for the entire data center, and 3 local MS's, two of which control 2 subtrees and one of them controls the top-level switches. The nodes marked with an (R) are the reference nodes and they are directly controlled by respective local MS's. The values of their configuration data are duplicated at the management side. H(SW_n) is the hash value for the configuration of SW_n. The aggregated hash values of the local management component are also stored at the global MS, which are AGG tuples. We describe the algorithm in detail below.

1) Algorithm Initialization: We assume that there is no external attack during the initialization process since this can be done before the assets go online, and we assume the management servers are not vulnerable. The groups are predefined by the administrators.

First, the algorithm chooses one reference node in a group, and such node is securely bound with the local MS for direct protection, which means more overhead. All the remaining assets in the same group verify integrity of their configuration by referring to the this node. The group central point and the radius r is stored in this node, and the hash values of the exact match parameters are also stored together. Such a node sends these standard values to the local MS, and each local MS also computes an aggregated hash value using hash tree over all of its cached groups' values. Legitimate updates of reference nodes are required to go through local MS and securely update these values accordingly. Each group standard value stored at the local MS is assigned a TTL (Time to live). The local MS will request the hash values to be recomputed if the TTL is expired. Algorithm 1 in the Appendix describes the process of initialization, and exact match parameters and range parameters are grouped, and the reference node is used to efficiently verify these two types of parameters.

2) Secure Update: For exact match parameters, we manage updates to the asset configuration data securely because all the assets within the same group need to update such parameters at the same time. By doing secure updates, the local MS always keeps the up to date values of each group and works as a gold standard. Before doing a secure update, a verification process is conducted to detect any suspect changes. In addition, updating the reference node must be followed by updates to the other nodes in the same group. Therefore, the updated reference node also broadcasts the updated configuration to other nodes in the group. When the new hash value of the updated exact match parameters is received by the local MS, it can record this update event as successful and log it.

In contrast, if a single node in a group updates some range parameters, then right after the node updates the parameter, another verification process will be conducted to check whether this update significantly change the configuration of this asset. If the update is minor, and the node still within r distance to the group central point, then the update is allowed, and such update will be reported to local MS. Otherwise, the verification process triggers an alert about this major update. If this major update is not malicious, then it is administrator's choice to assign the node to another existing group or regroup some assets. Group central points are not updated until regrouped.

For unique parameters, their values are always protected by secure updates. Each asset follows the steps the same as the ones of reference node updates for exact match parameters. These secure updates are expensive, so we do not expect too many critical parameters updating in each group.

Algorithm 2 in the Appendix depicts how a single secure update of a exact match parameter is conducted at both the local MS and the assets, and how the hash values are updated. It is important to make sure that the whole update process is atomic from attacker's perspective so that it is not possible to supply bogus values during the update. There are several ways to make the update secure and atomic. One possible way is to use PKI based encryption and authentication available in WBEM. Moreover, if the assets have a TPM like mechanism, a trusted channel can also be established to secure the update process, and software locks can be used to ensure atomicity.

3) Integrity Verification: To ensure the integrity of the configurations at assets, verification process is the most important step to detect any violation. Algorithm 3 in the Appendix shows the verifications of exact match parameters and range parameters. The verification within each group is efficient.

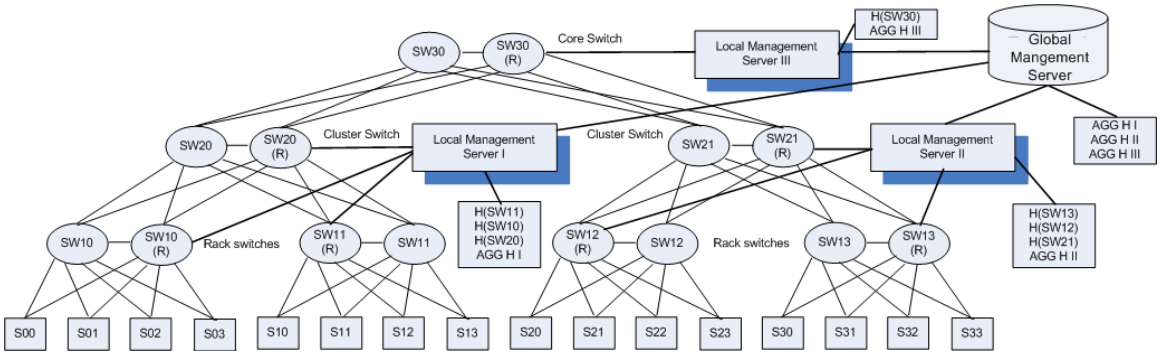

Fig. 2. Fat Tree Structure of Data Center Network

A verification process may be invoked because of an update request, or a TTL expiration (periodic check). The frequency of periodic checks (and the checks before updates) can be chosen according to the perceived security risks and its consequences. Upon detection of a misconfiguration or attack, the local MS can initiate a recovery process starting with the rediscovery of configuration parameters (like the one initiated on start-up).

Each reference node records the verification from the remaining nodes in the group. If any node does not invoke any verification for a time period more than a TTL, then an alert is generated. When doing the verification, different types of parameters are checked separately. For the exact match parameters, each node recompute the hash value over these parameters and sends it to the reference node, and the reference node compare it with the gold standards to detect any violation. For the range parameters, the corresponding vector is sent to the reference node, and the reference node compare whether the vector is within r distance to the group central point. If not, an alert is also generated. Unique parameters can also be checked with local MS since all the secure updates are logged by it, however, these are still expensive operations and are expected to be infrequent. By doing verifications at different levels periodically, integrity violations can be caught quickly before the data corruption has had any substantial impact on the data center operations.

4) Rotation of Reference Nodes: In order to avoid a reference node from being a target of persistent or brute-force attacks, we propose to rotate the reference node according to a randomized time schedule. Rotation can be invoked at the time a reference node needs to be verified. As the local MS selects a candidate node for the next reference node, the node must be first authenticated and verified by the local MS. Thus, for a short interval, there are two authenticated nodes within one group. The old node (currently reference) then sends a broadcast message to all the other nodes announcing the new reference node. After the local MS successfully records the rotation, the old reference node starts to behave as a normal node.

V. Application and Performance of Sentry

In this section, we discuss how Sentry can be adapted for data center network and servers, and evaluate its performance.

A. Securing Configuration Data

The Sentry mechanism is generic and takes advantage of the asset duplication. It is based the assumption that securely binding each node to the MS is expensive, whereas links among the same group of nodes are insecure but cheap. By rotating the reference node, the secure link and insecure links are changed so that a compromise is made harder for the attacker. This is a form of *moving target defense* mechanism in action, and it ensure that even if an insecure link is compromised, it will be detected after some time.

Notice that the secure update of exact match parameter is issued from the local MS, and any other valid updates are reported to the local MS; consequently, the server side will get the response and keep a log entry of the update. If a malicious intruder pretends to be a local MS and updates the configuration, there will be no record of the update on the real MS side, and malicious updates can be easily detected. Attacks trying to defeat the protection by overriding the address of the reference node or bypassing the check can also be detected. It is because that when the TTL expires and a verification is invoked, the reference node can not receive hash values from non-reference nodes, and as discussed above, the links binding reference nodes and the local MS can hardly be compromised. In addition, Legitimate and malicious modifications to the configuration data can be distinguished by the channels. Secure update must go through local MS, otherwise it is treated as malicious.

B. Overhead Analysis

Configurations of assets may need to be read or updated by the local MS from time to time. In the regular (i.e., unprotected) system, both reads and updates should be very efficient, and we assume reads and updates take the same amount of time. We also use the time slot of one such operation, denoted U_o as the basic time unit. We let R_r denote the frequency of these read/update accesses, and it is on a per time slot basis. With our scheme, the read time does not change since it is not secured. We assume that the overhead of a single secure update in our approach is O_1 times of U_o. Similarly, the overhead of a secure verification process is O_2 times of U_o.

For the TTL of the verification, we can choose the TTL expiration based on the time interval or based on the numbers

of read/update (access) events. As we choose to set the TTL based on the access events, we have a TTL expiration rate, denoted R_t. Here a $R_t = 0.1$ would indicate that the TTL expires after every 10 read/update events, and a verification process is needed then. Finally, we assume that the update rate is related to the read rate by a fixed ratio, α. Then we can have the percentage of utilization overhead as:

$$Overhead = R_r \times (1+\alpha \times (O_1\text{-}1)) + (R_r \times R_t) \times O_2$$

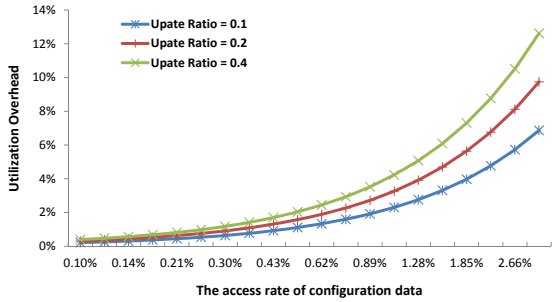

Fig. 3. Overhead analysis on a reference node

Fig. 3 shows some sample results on a reference node assuming $O_1 = 10$, $O_2 = 5$, and $R_t = 0.05$. The X-axis indicates the base rate of the read/update events. We consider access rate in the range of $0.001/U_o$ to $0.032/U_o$. Note that an access rate of $0.032/U_o$ means that there is one access every 1/0.032 or 32 time units, which is a fairly high rate for configuration read/update events. The Y-axis shows the percentage of overhead of our approach in terms of utilization. For the three different curves, we use the different update ratio α as 0.1, 0.2, 0.4 respectively. Since we expect update rate to be significantly less than the read rate, a maximum update ratio of 0.4 chosen here is quite pessimistic. As we can see with a update ratio of 0.4 and the access rate approaches the maximum, the overhead may goes up to 13%, and in most cases, the overhead will be less than 8%. It is obvious that the frequency of the update events and how we define the TTL are two major factors in the overall overhead.

C. Evaluation of LSH based Discrimination

We briefly study the performance of LSH based mechanism here. For this we used a set of configuration parameters from a real system and used it to generate experimental data. We used ten parameters, six of which were range parameters, and the rest of them were exact match parameters. In the first experiment, we generated two groups, and computed the group central point and the radius r for both of them (we set the radii of two groups very similar). Our goal was to evaluate the number of overlapping points of both groups when two groups are very similar. Therefore, we first only varied one range parameter and study the overlap as a function of the distance between the two groups.

Fig. 4 shows the result. The X-axis is the normalized distance between the two groups. For the normalization, we use use the sum of the radii of two groups, since at this distance there would be no overlap. The Y-axis shows the number of overlapping points that appear in both groups. In all the three runs shown, one group had 50 points but the other group had 50, 70, 90 points respectively. It is seen that the overlaps drop to near zero with only 50% normalized distance. This shows a good discrimination capability of LSH in detecting changes that might occur due to misconfiguration or deliberate corruption of the values.

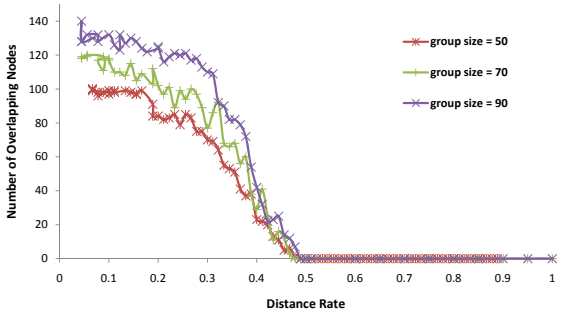

Fig. 4. LSH evaluation with two groups (different number of nodes)

Fig. 5 shows a different situation with 3 runs with one, two and 3 parameter changes. In this case, both groups have 50 points. It is seen that the number of parameters does not have substantial influence on the overlaps, and the LSH is able to pick up the perturbations quite well.

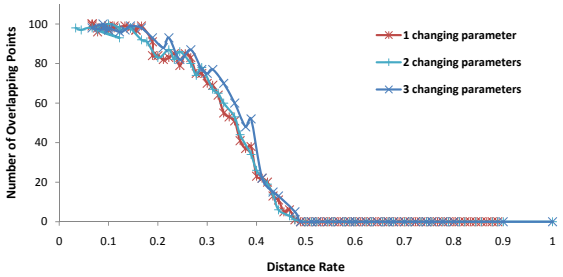

Fig. 5. LSH evaluation with two groups (different number of changing parameters)

Fig. 6 shows the overlap as a function of fractional distance when three groups are involved. For this case, each group has 50 points and we show results for two runs. For the first run, two of these are fixed and do not overlap, whereas the third group gradually moves away from the two fixed groups. In this case the greatest number of overlapping points is 100. For the second run, two of the groups move and one is fixed. For these cases, the fractional distance (plotted on the X-axis) is calculated as the sum of actual pairwise distance among three group centers divided by the sum of diameters of three groups. The results show that if the fractional distance exceeds 0.70, there is no overlap. Basically, these results are similar to those for 2 groups except for a somewhat slower decrease in the overlaps. These LSH results show that it is possible to group the assets in data centers efficiently and clearly, and there will be almost no false positive if they are properly grouped.

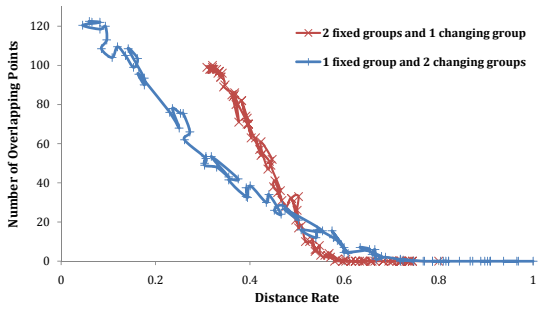

Fig. 6. LSH evaluation with three groups

VI. Related Work

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. References [12], [7] discuss the security challenges brought about by virtualization. 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 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 [14]. Reference [6] 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. The techniques considered in these and related research for isolating activities of VMs are based on the assumption that the basic mechanisms to achieve the isolation work reliably and are not under attack. In contrast, our concern in this paper is with attacks that might compromise these mechanisms and thereby fail to provide the desired isolation or protection. In particular, most network isolation schemes exploit VLANs but as pointed out later in the paper, VLAN configuration can be attacked in many ways.

In this paper we consider security of traditional configuration management. There have been some proposals to simplify network management by considering radically different schemes for partitioning the management responsibilities [1]. These schemes have not been adopted on commercial scale, and raise security issues of their own; hence we do not address them here. The aim of our work is also similar to Intrusion Detection Systems [20], which is to figure out any malicious changes and provide integrity to the protected data.

VII. Conclusions and Future Work

In this paper we examined the problem of securing configuration management (CM) within data centers. We discussed CM related vulnerabilities in a data center and noted that the web-services based management – although increasingly popular – can harbor attacks that can seriously disrupt data center operations. We then presented a novel mechanism called Sentry for securing the data center network that exploits the regular nature of data center network and the available redundancy.

This paper does not deliberately address the cloud computing environment since security in such environments must also address the issue of multiple parties being involved and a potential lack of trust between them. In fact, infrastructure as a service (IaaS) type of cloud access model would require splitting of configuration management between the infrastructure provider and the user. We intend to address this and other emerging complexities (e.g., security of virtual clusters) in our future work.

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.
[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. Butler, T. Farley, T. McDaniel, J. Rexford, "A Survey of BGP Security Issues and Solutions", to appear in Proc. of IEEE, 2010.
[5] "Common Information Model", Available at www.wbemsolutions.com/tutorials/CIM/cim-specification.html
[6] 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.
[7] Tal Garfinkel and Mendel Rosenblum, "When Virtual Is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments", USENIX Association, 2005
[8] P. Goyal, R. Mikkilineni, M. Ganti, "FCAPS in the business services fabric management", Proc. of 18th IEEE Intl. workshop on Enabling Technologies, 2009.
[9] 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.
[10] C. E. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing", IEEE Trans. on Computers, Vol 34, 1985.
[11] F. Palmieri and U. Fiore, "Enhanced security strategies for MPLS signaling", Journal of Networks, Vol 2, No. 5, Sept 2007.
[12] J.S. Reuben. A Survey on Virtual Machine Security. Helsinki University of Technology, 2007. Available at http://www.tml.tkk.fi/Publications/C/25/papers/Reuben_final.pdf
[13] S.A. Rouiller, "Virtual LAN security: weaknesses and countermeasures", available at uploads.askapache.com/2006/12/vlan-security-3.pdf
[14] 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.
[15] A. Stamos and S. Stender, "Attacking Web Services: The Next Generation of Vulnerable Enterprise Applications", Proc. of Defcon XIII.
[16] A. Striegel, "Security Issues in a Differentiated Services Internet", Proc. of HiPC workshop, 2002.
[17] 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
[18] Web service security specification, available at docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
[19] Web services secure conversation specification, available at specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf
[20] Stefan Axelson, "Intrusion Detection Systems: A Survey and Taxonomy", Technical Report, Chalmers University of Technology, 2000
[21] Gionis, Indyk and Motwani, "Similarity Search in High Dimensions via Hashing", Proc. of VLDB, 1999.

APPENDIX

Algorithm 1 Initialization Algorithm

Step1: Pick the reference nodes
for each local MS M_i **do**
  Form groups of assets based on their config. data
  for each group G controlled by M_i **do**
    Randomly select a ref. asset SW_r from group G
    SW_r broadcasts the results to other assets in G

Step2: Securely bind ref. nodes with local MS
for each local MS and ref. node pair (M_i,SW_r) **do**
  M_i asks for a hash value from SW_r
  SW_r computes hash value of its exact match config. data & sends H(SW_r) to M_i
for each local MS M_i **do**
  AGG-H(M_i)← H(H(SW_1), H(SW_2)...H(SW_n))

Step3: Compute group central point and radius r for each group
for each local MS M_i **do**
  M_i compute group central point and r,and send them to each reference node

Algorithm 2 Secure Update Algorithm

At local MS M_i side:
M_i sends update request via a secure channel to a ref. node SW_r
if M_i receives the update result back from SW_r **then**
  M_i gets the new hash value of H(SW_r) & replaces the old one
  M_i records this successful update to its log
  M_i recomputes AGG-H(M_i) and sends it to *global MS*

At the reference node SW_r side:
if SW_r receives update request through secure channel from M_i **then**
  SW_r applies the configuration update
  SW_r recomputes H(SW_r) and sends it back to M_i
  SW_r broadcasts the update request to other nodes in the group

Algorithm 3 Verification Algorithm

At local MS M_i side:
if a TTL of SW_r is expired or M_i requests update on SW_r **then**
  M_i sends verification request to SW_r
if M_i receives reply from SW_r **then**
  M_i compares the new value with the cached H(SW_r)
  if two values match **then**
    The verification is successful
else
  A misconfiguration or attack is detected
  M_i launches a recovery process

At the reference node SW_r side:
if A verification request is received from M_i **then**
  SW_r recomputes H(SW_r) over its exact match configuration data
  SW_r sends the new H(SW_r) and V_r to M_i
if a verification request is received from a regular node SW_n in the group **then**
  SW_r compares the received H(SW_n) with its own H(SW_r)
  if two values match **then**
    The verification of exact match parameters is successful
  else
    A misconfiguration or attack is detected
    SW_r sends the correct configuration data to SW_n
  SW_r compares V_r received with the group central point
  if V_r is less than r from the group central point **then**
    The verification of range parameters is successful
  else
    A misconfiguration or attack is detected
    SW_r reports the alert to M_i

At the non-reference node SW_n side:
if the TTL of SW_n is expired **then**
  SW_n recomputes H(SW_n) over its exact match configuration data
  SW_n recomputes the vector V_r over its range configuration data
  SW_n sends H(SW_n) and V_r to SW_r and require verification