Retrofitting the IBM POWER Hypervisor to Support Mandatory Access Control

Enriquillo Valdez Reiner Sailer Ronald Perez
IBM T. J. Watson Research Center, Hawthorne, NY 10532
{rvaldez, sailer, ronpz}@us.ibm.com

Abstract

Server virtualization more readily enables the collocation of disparate workloads on a shared physical platform. When employed on systems across a data center, the result can be a dramatic increase in server utilization and a decrease in overall power, cooling and floor space requirements. However, in an environment where workloads share the underlying platforms, achieving other desirable workload goals, such as availability and security, becomes a challenge. In particular, enforcing isolation between workloads in a large, dynamic, and virtualized data center requires strong yet easily configurable controls on the sharing of resources at the virtualization layer. Commercial hypervisors usually offer reasonable isolation of individual virtual machines (VMs). However, on hypervisor-based platforms, one cannot currently define a single policy that automatically enforces restrictions on the sharing of resources between multiple VMs or request an air gap between workloads.

In this paper, we describe the design and implementation of a Hypervisor-based Mandatory Access Control (MAC) that achieves policy-driven distributed workload isolation for the IBM Power Hypervisor (PHYP). We discuss our experiences and lessons learned and examine the implications and trade-offs involved in providing MAC on a production-level, commercially-available hypervisor. Our goal is to simplify the security management of data centers through centralized security management and policy-driven distributed access control and data protection.

1. Introduction

A workload consisting of tasks and services can be distributed among a set of virtual machines (VMs) executing on a single or across multiple platforms for reliability and efficiency reasons. In this environment, hypervisors traditionally isolate individual VMs and enable the sharing of resources on the platform. The sharing properties between VMs in this environment rely largely on the discretionary decisions of administrators to correctly configure the resource sharing among VMs (e.g., network, storage) according to the overall workload isolation goals.

This environment, however, does not have a formal basis for expressing the controlled sharing of resources or platform to guarantee workload isolation. This lack of formality makes it hard to reason about the isolation provided to workloads. Data center operators, for example, want to ensure that the sharing of resources is based on the type of the workload and that selected designated workloads are prevented from executing at the same time on the same platform. Consequently, in multi-tenant computing or data center environments, it becomes difficult to keep track of the resources and distributed workloads and to establish proofs of compliance of workload isolation through audit logs.

In this paper, we investigate retrofitting the IBM POWER Hypervisor (PHYP) [2] with sHype mandatory access control architecture [24] to enable policy-driven workload protection. PHYP is a commercial-grade hypervisor that offers the functionality and resources that modern virtualization environments demand. It provides isolation guarantees [3] to virtual machines called Logical Partitions (LPAR). PHYP prevents programs running in an LPAR from affecting other programs running in other LPARs, isolates LPAR memories, and allows exclusive physical device assignments to LPARs by administrators via a stand-alone Hardware Management Console (HMC). Throughout this paper, we use the terms “LPAR” and “VM” interchangeably.

For PHYP, we provide a workload protection mechanism to mediate resource assignment (access) by mandatory access control (MAC) on LPARs. This MAC enforcement is independent of the LPARs and does not require their co-operation. Our workload protection mechanism is based on a simple security policy that can be applied across heterogeneous platforms to enforce consistent isolation properties for a virtual data center. Adding MAC to PHYP allows administrators to be confined to managing a single

workload and its resources. This provides a safety net by preventing information flow violation due to improper resource configuration by administrators. Additional benefits of MAC enforcement in PHYP include supporting least privilege by confining workload types to a platform, establishing the basis for safe object re-use, and providing easier proofs of isolation and anti-collocation guarantees. Our approach allows for security policies to be centrally managed (i.e., authored) and enforced locally on the distributed platforms. This can simplify the security management of large scale computing environments considerably.

Other hypervisor MAC approaches, such as KVM/370 [9][1] or VAX/VMM [15], aimed for high-assurance and required implementing a new hypervisor or large intrusive changes to the existing hypervisor. Our approach aims to minimize intrusive changes and depends on the core isolation capabilities and their assurance in the underlying hypervisor (PHYP). Although this work is similar to the sHype MAC architecture in the Xen open source hypervisor [26], the challenge of this work was to retrofit PHYP without considerably affecting the existing infrastructure and software. We succeed in applying a new approach where we mediate configuration commands only instead of runtime commands. This approach is non-intrusive to the critical path and yields no access control-related runtime performance overhead, since access control is enforced during configuration time. There will be overhead induced by the way we can assign resources, configure networks, and also by running multiple Virtual I/O Servers to securely share resources among VMs of a workload.

Initially, we explored implementing multi-level security (MLS) for the IBM research hypervisor [10] in the sHype access control framework, but found that our simple policy model, though less expressive, maps better to the virtual machine monitor (VMM) abstraction and operations. The VMM's coarse-grained operations, such as allocation of resources to VMs and enablement of communication between VMs, are intrinsically bi-directional. We find that our simple sharing policy fits better onto the virtualization abstraction because its bi-directional policies do not require distinguishing read from write operations between subjects and objects. Consequently, our simple policy yields a less-intrusive implementation which makes the adoption of mandatory security in a commercial environment more viable. If finer-grain controls are necessary, we propose a layering of access controls within Guard VMs as described in [21].

In this paper we discuss our practical experience in applying the sHype access control architecture to the PHYP environment. Section 2 provides an overview of the PHYP platform. Section 3 presents our MAC design and supported security policies. Section 4 discusses the implementation of MAC on PHYP. Section 5 reviews lessons learned in applying our approach. Section 6 presents future work. Section 7 reviews related work. Finally, Section 8 summarizes our results.

[1] Not to be confused with the new KVM [12].

2. PHYP Platform Overview

PHYP is the virtualization engine for IBM's PowerPC-based System i/p platforms. System i/p platforms are targeted for corporate and data center environments. Figure 1 shows the main components of a managed Power Hypervisor platform.

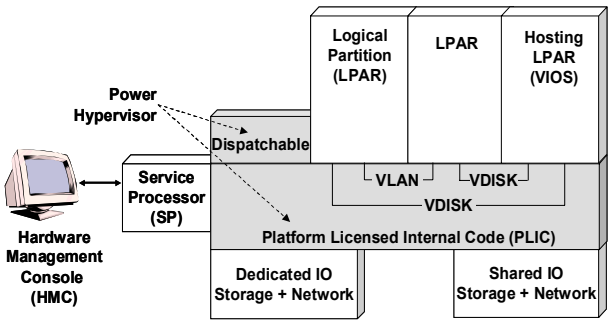

Figure 1. Power Hypervisor Architecture

The Hardware Management Console (HMC) [20] is a dedicated PC that runs the management application. It provides the interface for configuring and managing the platform. The HMC communicates configuration information via the service processor to PHYP over a dedicated management network using the Secure Sockets Layer (SSL) protocol and password protection. The service processor is an independent subsystem that performs system diagnostics and maintains platform configuration information.

PHYP consists of two parts: a non-blocking interrupt driven layer, called Platform Licensed Internal Code (PLIC), and a multitasking kernel, called Dispatchable PHYP. PLIC performs time critical operations required for virtualization. It enforces, for example, the partitioned environment. PLIC maintains the hardware page tables for translating an LPAR memory address into a physical address. It validates an LPAR's access to its hardware page table entry. This prevents an LPAR from accessing the memory of another LPAR. Similarly, it maintains a Translation Control Entry (TCE) table, an IO memory map unit, which is used to translate addresses generated by IO devices to physical memory assigned to LPARs. An

administrator at the HMC must first assign ownership of a physical IO device to an LPAR before the LPAR is permitted to map a portion of its memory to the TCE entry associated with the device.

Dispatchable PHYP executes as a hidden LPAR and provides non-critical platform services. One of its main duties is to process messages from the HMC and services processor. The HMC provides Dispatchable PHYP with LPAR configuration data. Dispatchable PHYP is responsible for processing and maintaining configuration data on the platform. It participates in every configuration update even in the processing of Dynamic LPAR operations where resources are added or removed on running LPARs. Dispatchable PHYP is also involved in directing the startup and terminations of LPARs. Additionally, it provides virtual service processors to the LPAR. These mirror real services processors and are used to maintain LPAR state information.

PHYP employs the para-virtualization approach where operating systems have embedded hypervisor calls for requesting PLIC virtualization services. Accordingly, PHYP can run any operating system developed using the PowerPC Architecture Platform Reference Specification [16] for interfacing with the PowerPC platform. Figure 1 also shows two client LPARs and a hosting LPAR, Virtual IO Server (VIOS). Dedicated IO can be assigned to any LPAR on the platform. Only the VIOS can be assigned shared IO. Hence, the VIOS can be configured to provide virtual storage or Ethernet to LPARs that lack the physical resources. Our workload protection focuses on mediating the sharing of the platform and resources (physical or virtual) and relies on PHYP to ensure isolation of platform resources such as CPU, memory, and physical IO devices [2].

Because PHYP is closed-source software, there are limits on the information that can be publicly disclosed about PHYP. Internally at IBM, there has been extensive review of the PHYP source code. With respect to design, PHYP architects follow the overriding design principle of only performing tasks in Dispatchable PHYP or in PLIC that can not be performed elsewhere in the software stack (i.e., application space, OS kernel, or VIOS). Currently, PHYP for POWER6 processors is under Common Criteria Evaluation EAL4 [7].

3. PHYP MAC Design

Three goals drive the design of sHype MAC enforcement in PHYP. The first goal is to implement MAC that confines workloads in case of an LPAR compromise and prevents administrators from configuration mistakes that would create sharing between workloads that are intended to be confined from each other. For this reason, our design does not assume cooperation of user LPARs for providing isolation. The second goal is to provide a non-intrusive design. Such a design minimizes impact on the PHYP code base, which in turn increases the likelihood of the PHYP Design and Development Team accepting those MAC extensions. The third goal is to have negligible performance overhead so that performance will not become a hurdle for its acceptance by customers in high-utilization environments.

We address the first goal by enforcing Mandatory Access Control (MAC) on configuration commands that change the assignment of resources to LPARs. We define a MAC security policy to specify the access to resources by LPARs based on security labels, which are attached as protected meta-information to LPARs and resources. The MAC security policy can be installed administratively on the platform, preferably by a security officer. Only the security officer will be able to effect changes on the installed policy. We address the second goal by minimizing changes and additions to the PHYP code base. We support the third goal of minimizing the performance overhead by performing access checks at configuration time and not during every run-time access of an LPAR to one of its configured resources.

3.1. Reference Monitor Approach

The design of our PHYP MAC enforcement is based on the principles of the reference monitor approach as introduced by Anderson [1]. In this approach, a subject's access to an object is mediated by a reference monitor. The security of this approach rests on three fundamental requirements. First, the reference monitor cannot be bypassed. It is always invoked when a subject accesses an object. Second, it is tamperproof. Subjects cannot alter its functionality. Third, the reference monitor is small enough to allow its correctness to be easily verified.

In our case, a subject is an LPAR and an object can be an LPAR or a resource. The reference monitor approach requires security labels to be assigned to subjects and objects. When an LPAR accesses an object, the reference monitor is invoked. The reference monitor allows or denies access based on the security labels and the security policy being enforced.

For PHYP, the MAC design consists of three components: an access control policy, access control module (ACM), and security hooks. The access control policy defines the policy to be enforced on the platform, including the security labels which can be

assigned to LPARs and resources. The policy is loaded into the ACM, which is responsible for providing the access control decisions based on security labels. Security hooks are guarded method invocations that request security access decisions from the ACM. The ACM and the security hooks constitute the reference monitor validation mechanism. The separation between ACM and security hooks follows a well established principle of separating policy from enforcement. Enforcing MAC on PHYP requires determining the mediation points in the platform.

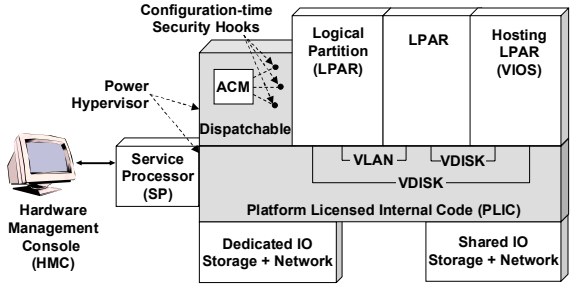

Figure 2. Mediating configuration commands

We implement mandatory access control in Dispatchable PHYP to avoid changes to performance-critical paths in PLIC and to minimize the intrusiveness of our implementation, see Figure 2. We leverage PHYP's centralized configuration setup to apply MAC before the LPARs are even powered on, at the time when resources are assigned (configured). As a result, MAC enforcement is performed during configuration updates. A configuration update is only accepted by PHYP, specifically Dispatchable PHYP, if it passes the access control checks. In our design, when the HMC submits configuration requests to Dispatchable PHYP, PHYP accepts or denies the requests based on the security labels of the subjects and objects that are involved in the configuration update.

This design successfully meets the first and second requirements of the reference monitor: The ACM resides in PHYP and thus cannot be tampered with by user LPARs. PHYP is the highest privileged code and is protected against the LPARs running on the platform. Since all configuration requests must go through Dispatchable PHYP, a bypassing of the MAC enforcement is not possible. This provides strong security guarantees since enforcement takes place when resources are configured to LPARs.

Since we favor a non-intrusive design over a small trusted computing base (TCB) [8] in our commercial environment, we do not optimize the third reference monitor requirement and accept as a trusted computing base the PHYP hypervisor and management environment on top of the system hardware. Although the ACM is a small module and there are only a few security hook instrumentations, we must consider other parts of the PHYP code base and hardware that contribute to the security administration and protection as part of the reference monitor. The TCB of the retrofitted PHYP consists of the HMC, SP, and Dispatchable, PLIC as well as the hardware that enables the privileged hypervisor state and physical isolation capabilities. Addressing the third principle fully would require partitioning PHYP's TCB into security and non-security parts (this would be similar to the KVM/370 [9] retrofitting effort). However, this in turn would violate our overriding design requirement to minimize intrusiveness of the protection mechanism on PHYP.

3.2. Simple Policies

We support two simple orthogonal security policies to govern authorization and resources allocations on PHYP: Simple Type Enforcement (STE) and Chinese Wall Enforcement (CHW).

The simple type enforcement policy enforces restrictions on the communication and resource sharing between LPARs or between an LPAR and a resource based on the STE type (e.g., color) associated with the LPARs or resources. The STE policy specifies that LPARs can only communicate with each other if they have a common STE type, i.e., both LPARs must have assigned at least one type in common in their security labels. Similarly, an LPAR is permitted access to a resource, if the LPAR and the resource have a common STE type. Typically, resources and LPARs have a single STE type.

Due to limited physical resources on a platform, some platform configurations employ a VIOS partition to enable the sharing of hardware resources, such as a storage and network devices, among multiple client LPARs. Such configurations have implications to the STE type assignments and the reference monitor's TCB. If a VIOS provides resources to LPARs with different STE types, then the VIOS must be assigned multiple STE types corresponding with the STE types of the client LPARs. Additionally, the VIOS must be MAC aware since it must map its PHYP configurable resources to internal resource mapping based on the STE types. Since a multi-STE typed VIOS must mediate LPAR accesses to its internal resources based on the STE type, it becomes part of the reference

monitor's TCB for those types it is assigned (c.f., MAC-Domains in [24]). To avoid adding a full-sized VIOS to the TCB, we can employ multiple single STE typed VIOSs to service only single STE typed client LPARs. For scarce hardware, we can deploy a minimized VIOS to share such hardware more safely among differently labeled LPARs.

In contrast to STE, the Chinese Wall policy controls which workloads can run on the same platform at the same time and which cannot. If workload types A and B are designated as conflicting then, as long as an "A" typed LPAR executes, no "B" typed LPAR is allowed to execute on the platform and vice versa. Chinese Wall types that shall not be collocated define a so called conflict set and LPARs with workload types that are in a common conflict set will run mutually exclusive on the platform. This approximates an air gap between conflicting workloads assuming that the platform does not keep state of LPARs once they stop executing. In [13], the authors discuss an approach that leverages the Chinese Wall collocation restrictions to reduce covert channels risks between specific workloads.

3.3. Operations and Resources under MAC

To enforce collocation restrictions with the Chinese Wall security policy component, we control the assignment of the bootable state of an LPAR based on its security label. An LPAR set to the bootable state is allowed to be powered on the platform without additional policy checks. Therefore, once an LPAR with conflicting CHW types is set to bootable, other LPARs with conflicting types are not allowed to be set to bootable. In order to determine when the other LPARs from the conflict set can be eventually set to bootable, we must track the bootable LPARs according to the CHW policy's conflict sets.

To enforce the STE policy component, we control the configuration of the following virtual and physical peripheral resources to LPARs based on security labels assigned to LPARs and resources:

- vSCSI, vTTY - are virtual resources that allow interaction between a client LPAR and a server LPAR (e.g., VIOS). The server LPAR provides services such as network connectivity or virtual storage to client LPARs.
- vEthernet – is a virtual resource that enables inter-LPAR communication. LPARs can communicate with each other if their Logical vEthernet adapters are assigned to the same VLAN ID.
- An IO pool – is a group of IO devices that can be shared by a group LPARs without requiring active HMC involvement.
- VLAN Switch – is an internal switch that enables LPARs to communicate based on their VLAN membership.
- Host Ethernet Adapter (HEA) – provides LPARs direct high speed access to the network via logical ports without going through a VIOS.
- Physical IO devices – allows direct access of physical IO devices by LPARs. An IO device is exclusively owned and used by a single LPAR.

Next, we discuss the assignment of security labels and how they are used to enforce STE and CHW policies. In our environment, we view a security label as a container for CHW and STE types.

Label assignment. We assign security labels—defined as part of the platform security policy—to all LPARs and resources before they can become active:

- A security label for an LPAR contains CHW and STE security type attributes. A device on an LPAR's virtual bus is assigned only a single STE type. This allows a multi-STE typed LPAR to distinguish the security type of the connection.
- User LPARs are usually assigned only a single STE type since we do not trust them to keep different types confined. LPARs that are assigned multiple STE types can implement sharing of hardware or create controlled information flow between user LPARs of different STE types. However, multi-STE typed LPARs must be trusted to confine the types and only permit selective information flow if desired. If a multi-STE typed LPAR is compromised or untrusted, no confinement guarantees hold for those STE types that are assigned to such an LPAR. Consequently the least privilege principle should be applied when assigning STE types to trusted LPARs.
- A security label for a VLAN, IO pool, HEA, or physical IO device has exactly one STE type. We label the physical slot location of an IO device since the device is not MAC aware.
- Shared resources can only be assigned a single STE type, since those resources otherwise could be accessed by different STE typed LPARs and information flow through those shared resources would violate the type confinement requirements. Chinese Wall types do not apply to resources.

A complex hardware device can be assigned more than one STE type if it is composed of multiple isolated hardware components, access to which can be individually controlled by PHYP. In this case, however, an isolated sub-component is still assigned a single STE Type. Although not a physical hardware

device, PHYP's internal Virtual Ethernet Switch supports the IEEE 802.1Q VLAN standard which provides VLAN isolation. In the PHYP Ethernet Switch case, we assign each VLAN ID a single STE type and check this type when the VLAN ID is assigned to the LPAR's vEthernet adapter.

MAC enforcement. On the PHYP platform, MAC enforcement restricts the collocation of LPARs and the assignment of resources to LPARs according to the platform security policy. An assignment of the bootable state to an LPAR is permitted, if the LPAR's CHW type does not conflict with any LPARs that are already in the bootable state. This prevents the activation of conflicting LPARs.

The configuration of a client LPAR adapter to a server LPAR adapter is permitted, if the client LPAR and server LPAR share an STE type in their security labels. This covers vSCSI and vTTY adapter configurations. A multi-STE typed server (or client) is able to determine the STE type assignment of its adapter and enforce the confinement against adapters of other STE types. An assignment of an LPAR to a resource is permitted, if the LPAR includes the resource's STE type in its label. A resource in our environment could be an IO POOL, VLAN, physical slot, or a Logical Port from a HEA.

3.4. An example of MAC enforcement

We review MAC operations on the platform illustrated in Figure 3 to show how MAC enforcement works on PHYP platforms. This figure shows a managed platform with a hosting LPAR (VIOS) and two client LPARs. The VIOS owns the physical hardware disk and serves virtual disks to the client LPARs.

A platform security officer defines STE types {green, red, service} and CHW types {green, red, service}. For simplicity reasons, the figure does not differentiate between CHW and STE types. For the CHW types, a conflict set {green, red} is defined. The security officer created Red, Green, Res, and Service security labels to group the types as shown on the left in Figure 3.

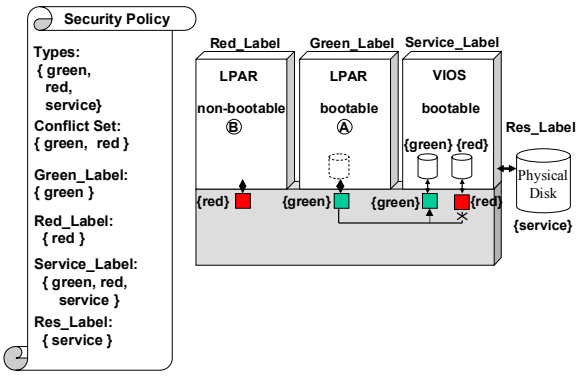

Figure 3. Labeled LPARs and resources

STE security check. At configuration time, an administrator assigns the Service Label to the VIOS LPAR and the Res Label to the physical disk. Since the VIOS and physical disk share the service STE type, the administrator can then assign the physical disk to the VIOS. To provide virtual disks from the physical disk, the VIOS is configured with two server vSCSI adapters. The VIOS has tagged one vSCSI adapter with STE {green} and the other with STE {red}. The administrator also assigns the Green Label to client LPAR, *LPAR_A*, and the Red Label to the other client LPAR, *LPAR_B*. Since each client LPAR has one STE type in its security label, its client virtual SCSI adapter automatically inherits its STE type accordingly.

Later during configuration, if an administrator tries to configure LPAR_A's client adapter to the VIOS's red vSCSI adapter, PHYP denies the assignment since the adapters do not share an STE type. In contrast, PHYP accepts the assignment of LPAR_A's client adapter to the VIOS's green vSCSI adapter. The VIOS is required to keep the virtual disks safely separate and connected to the correctly labeled server VSCI adapters.

CHW security check. While LPAR_A is the first client activated, i.e., set to bootable state, any later setting bootable of LPAR_B is rejected by PHYP. This is due to the client LPARs having conflicting CHW types in their security labels. Since the activation of an LPAR is not a configuration action (uncontrolled), we control the bootable flag in the LPAR configuration to ensure that the anti-collocation rules are enforced. The implementation was driven by simplicity in enforcing the policy at configuration time and a small loss in granularity through this 'pessimistic' interpretation of the bootable flag instead of the running state is accepted.

4. Implementation

For our proof of concept, we integrated sHype MAC support into Dispatchable PHYP. We employed debugging tools to load a policy, assign security labels, and test policy enforcement. Our prototype implementation focused on policy representation and on extensions to PHYP to support MAC. Note that the MAC implementation is currently not part of any production level release of PHYP.

4.1. Policy Representation

To facilitate policy processing and understandability, we employ three different representations of a security policy: XML, mapping and binary representation. The latter two representations are used for enforcing MAC on PHYP based systems.

Expressing a policy as an XML document provides for a standardized representation of labels and conflict sets. This representation allows users and automated tools to easily author and update a policy that is independent of platform specifics. From the XML policy representation, we derive the mapping and binary policy representations.

The binary policy is a low-level representation designed to optimize policy processing in PHYP. This representation is used directly within the ACM to perform the access control decision. The mapping policy is an intermediate representation that links the XML and binary representations. It contains the mapping of symbolic label and type names to the low level representation contained in the binary policy. The mapping representation is most useful for policy enforcement within VIOS partitions.

4.2. PHYP Extensions

To implement the sHype access control architecture, we extended Dispatchable PHYP with our ACM implementation and instrumented the configuration processing module with security hooks that automatically invoke the ACM's decision API.

The ACM is the access mediation component that encapsulates the STE and CHW policy engines. It provides two major functions. First it instantiates the policy engines based on the loaded policy. Second it delegates access decisions to the respective policy engine. Every ACM decision method returns a security access decision and a status return code. The status return code indicates if a processing error occurred during the security access check.

Our ACM implementation is written in C++ and is about 2500 lines of code, including the code for the STE and CHW policy engines. Since the current PHYP LPAR configuration profiles do not support security label entries, our current ACM implementation also includes support code for maintaining the associations of security labels with LPARs and resources. The ACM decision API is invoked with the subject and object identifiers, such as LPAR ID, IO pool ID, VLAN Switch and VLAN ID, or physical slot location ID. These identifiers are used to retrieve label information from the ACM label association cache. Adding ACM decision support for the Host-Ethernet-Adapter (HEA) is ongoing work because this hardware became recently available.

The security hooks are responsible for triggering and enforcing access control decisions. We implemented six security hooks to control all configuration functions in Dispatchable PHYP. These are guarded calls to ACM decision methods and are located in the PHYP module where Dispatchable PHYP processes configuration requests from the HMC. Depending on the resource being configured, a security hook invokes the appropriate ACM decision method. If the method's return status is OK and the decision is PERMIT, Dispatchable PHYP continues its normal processing of the configuration request. Otherwise, it returns an error for the configuration request.

4.3. Testing

We utilized proprietary debugging tools from the PHYP Design and Development Team for testing ACM functionality on the platform. As previously described, the access control decisions are triggered during the processing of configuration requests in Dispatchable PHYP. For testing the decision functions, we depended on the capability to set a policy and assign labels to LPARs and resources on a PHYP platform.

To install a policy on a platform, we used a PHYP debugging utility to load our binary and mapping policy representation files directly into Dispatchable PHYP memory. We also leveraged PHYP Development Team's native macro feature, which allows Dispatchable PHYP to be extended with a code module written in C++. A native macro can be invoked via the PHYP debugging interface. Leveraging this macro feature, we implemented a *cfacm* macro to drive MAC operations on the platform. The *cfacm* macro provides operations such as activating a policy, assigning security labels to LPARs and resources, and invoking the ACM decision API. Working on a

development release of PHYP, we established a data connection via telnet to Dispatchable PHYP using the Virtual Server debugging interface. This provided a command line interface for invoking the *cfacm* macro. This feature is available only in development releases.

We tested our MAC enforcement on a System p model 520 hardware using the latest PHYP development release. We loaded the binary policy from our example in Figure 3 which consisted of 192 bytes into Dispatchable PHYP. Note that the binary policy grows linear with the number of labels and types. Using the *cfacm* macro we manually tested ACM's API for access control decisions. We also tested activation of LPARs with conflicting CHW types using the HMC. Our experiments generated the expected results. Access to resources was allowed or denied depending on the security label assignments.

5. Lessons Learned

In providing MAC for PHYP, we learned two major lessons. First, we can implement MAC non-intrusively on the commercial-grade POWER Hypervisor as a result of the way resources are configured exclusively through Dispatchable PHYP and allocated in PLIC during runtime. Second, simplifying the notion of a security label helped non-security people with their understanding and application of a security policy.

Non-intrusive design and implementation: Initially, it was proposed to provide MAC enforcement on the HMC. This is an attractive solution because it requires no changes to the PHYP platform. Since the HMC already provides LPAR configuration to PHYP, it can easily ensure that an LPAR configuration does not violate the access control policy. However, this approach does not scale for managed platforms. In a PHYP platform, multiple HMCs are allowed to manage a single PHYP platform. Enforcing MAC on the HMC would require coordination and coherence among the HMCs. This coordination to synchronize their MAC operations overly complicates the manageability of MAC. Thus, having MAC enforcement in PHYP simplifies MAC processing, eliminates the dependency on the HMC, and ensures consistent enforcement of a central security policy independently of the HMC.

Ideally the HMC can contribute to MAC enforcement on the platform by determining the validity of the MAC operations before sending configuration requests to PHYP. The HMC can ensure that HMC users (administrators) do not try to assign incompatible resources to LPARs by listing only resources for assignment to LPARs based on their security labels. The HMC proves useful for authoring security policies and managing label associations as well as providing configuration guidance to users. It is less useful for single-handedly enforcing the policy.

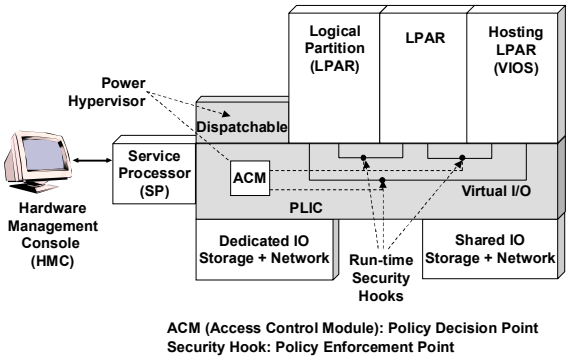

Figure 4. Initial PHYP MAC Design

Once we decided that MAC enforcement should reside on the PHYP platform, we thought that the straightforward approach would be to enforce the policy at run-time when LPARs bind to resources or access resources. Consequently, our initial implementation placed the ACM and security hooks directly into PLIC, see Figure 4. Performing access control decisions at resource binding or access time minimizes the code path between the time of access check and the time of resource access. Thus, this implementation depends on less code for correctness. It also allows for revocation and re-labeling through callbacks. However, this implementation is characterized by having security hooks distributed throughout many PLIC modules and leads to a more intrusive implementation. The sensitivity of PLIC on platform performance and the intrusiveness of this solution led to rejection of this approach.

Our current implementation, discussed in Section 3.1, instruments Dispatchable PHYP with the ACM and security hooks. It centralizes policy processing and decisions into one module, incurs no runtime cost, and does not impact PLIC. This approach is driven by our observation that the runtime setup is confined through the configuration settings in PHYP. However, this approach does not support automatic revocation of resource access based on re-labeling.

Our final lesson was that non-security people have difficulties in understanding a security policy when security labels are defined in terms of types and set operations. Using the color metaphor for a security label provided a good way to visualize the concept to non-security people. Additionally, using the same types in both STE and CHW policies further simplified the understanding the MAC policy.

6. Future Work

Future work includes extending MAC support to PHYP management applications and making the VIOSs (hosting LPARs) MAC aware.

Supporting MAC on the PHYP platform requires adding MAC enablement features into the PHYP management applications, including the HMC and the Integrated Virtualization manager (IVM) [11]. IVM runs on the VIOS and provides a subset of HMC management features. The HMC and IVM must be extended to create a MAC policy, assign labels to LPARs and resources, and load and update MAC policy into Dispatchable PHYP.

Additionally, the management applications can support safe object reuse by requiring the backup and cleaning of (virtual) storage resources before they can be re-labeled and re-used for a different workload. Similarly, removing a label from a resource can trigger a process that ensures that the resource is backed-up and cleaned.

To be able to scale the authorization required to manage the MAC PHYP platform, we intend to separate administrative duties in the management applications. Platform management can be divided between a security officer and non-security administrators. The security officer would be responsible for setting the security policy and labeling LPARs and resources. The non-security administrator would be responsible for performing non-security management tasks on the platform.

Finally, VIOSs that are multi-STE typed must be MAC aware, since they need to enforce the MAC policy on virtual resources exported to client LPARs. We plan to include hooks into the VIOS to implement the access controls required to enforce the policy.

7. Related Work

Hypervisors can be divided into two categories: isolation and sharing hypervisors [14]. Isolation hypervisors do not allow the sharing of resources between LPARs. Rushby [23] formalizes the necessary conditions for ensuing isolation among users on a single machine. Kelem and Feiertag [17] extend Rushby's separation model to VMs executing on the same platform. An example of a separation model implementation is NetTop [19], which isolates VMs based on their sensitivity levels on a platform, but allows VMs to connect to networks. Another example of a pure isolation hypervisor is the IBM PR/SM system [6].

Our work falls in the sharing hypervisor[2] category. The Xen hypervisor's sHype MAC implementation [24] for para-virtualized VMs is closely related to our work. Although Xen's and PHYP's MAC realization provide the same type of policy enforcement, their implementations differ considerably. Xen's and PHYP's configuration and deployment schemes employ different approaches for implementing MAC. MAC enforcement for Xen must be performed partly in the interrupt driven hypervisor layer during runtime. In contrast, PHYP's MAC enforcement is performed during configuration time only. Ultimately, we want to support MAC enforcement across multiple PHYP platforms; the solution presented in [18] for sHype on Xen applies to sHype on PHYP as well.

Other sharing hypervisors with security kernels include VAX VMM [15] and KVM/370 [9]. Both VAX VMM and KVM/370 were developed for high-assurance and required implementing a new hypervisor or large and intrusive changes to the existing hypervisor to achieve their security goals. They provide multi-level security models. Our sHype MAC retrofit for PHYP is non-intrusive and aims at the assurance that the base hypervisor is designed for (usually enterprise-level assurance) and provides a simple security policy model that is easy to understand for administrators in virtualized environments. Additionally, the sHype security checks in PHYP do not incur any runtime overhead since they are performed at configuration time.

Further, security enhancements to Multics [25] are related to our work. The authors describe how users' access to classified information at the OS layer is controlled by using a restricted multi-level security mode. In contrast, we restrict VM access to resources at the hypervisor layer using a simple and platform-independent policy model. We note that the Multics enhancement work also describes administrative and physical safeguards (e.g., separation of administrative duties, secure terminal) which are applicable to PHYP management applications.

8. Conclusion

In this paper, we show how the sHype mandatory access control architecture can be implemented for the commercial-grade PHYP hypervisor with minimal impact on the code base and performance. We consider the limitations and tradeoffs of our approach with respect to meeting the three fundamental principles of the reference monitor. One key observation is that the reference monitor introduced in the early seventies still remains relevant today for mediating access to

[2] Of course, the original sharing hypervisor is the CP-67/CMS [22].

resources. Another result of our work is that we can provide simple and robust protection statements to customers about their workloads using simple security polices. This has the potential to simplify the management and harden the security of the platforms.

9. Acknowledgements

The authors would like to thank the IBM POWER Design and Development Team for providing access to PHYP information and platforms. In particular, we want to thank Bill Armstrong, Pete Heyrman, Bryan Logan, Kyle Lucke, Amartey Pearson, David Larson, and David Engebretsen for their generous assistance. We would also like to thank Paul Karger for his comments on previous MAC work on hypervisors.

10. References

[1] J. P. Anderson. Computer Security Technology Planning Study. ESD-TR-73-51, Vols. I and II, Air Force Electronic Division Systems, Hanscom AFB, Bedford, MA, Oct. 1972.

[2] W. J. Armstrong, R. L. Arndt, D. C. Boutcher, R. G. Kovacs, D. Larson, K. A. Lucke, N. Nayar, and R. C. Swanberg. Advanced Virtualization Capabilities of POWER5 Systems. *IBM Journal of Research and Development,* Vol. 49, No. 4/5, July/Sept. 2005.

[3] B. Armstrong, S. Bade, D. Boutcher, C. DeRobertis, T. Mathews, and A. McLaughlin. LPAR Security on POWER5 Processor-based Systems, Sept. 2007. URL: http://www.ibm.com/systems/p/hardware/whitepapers/lpar_security.pdf.

[4] W. E. Boebert and R. Y. Kain. A Practical Alternative to Hierarchical Integrity Policies. *8th National Computer Security Conference*, 1985.

[5] D. F. C. Brewer and M. J. Nash. The Chinese Wall Security Policy. In *Proc. IEEE Symposium on Security and Privacy*, pp. 206-214, May 1989.

[6] Certification Report for Processor Resource/System Manger (PR/SM) for the IBM eServer zSeries 900, BSI-DSZ-CC-0179-2003, Bundesamt fur Sicherheit in der Informationstechnik, Bonn, Germany, 7 Feb. 2003. URL:http://www.commoncriteriaportal.org/public/files/epfiles/0179a.pdf.

[7] Common Criteria Evaluation and Validation Scheme. URL:http://niap.bahialab.com/cc-scheme/in_evaluation.cfm.

[8] Department of Defense. *Trusted Computer System Evaluation Criteria (Orange Book)*, DoD 5200.28-STD, 1985.

[9] B. D. Gold, R. R. Linde, and P. F. Cudney. KVM/370 in Retrospect. In *Proc. IEEE Symposium on Security and Privacy*, 1984.

[10] IBM Research. The Research Hypervisor – A Multi-Platform, Multi-Purpose Research Hypervisor. URL:http://www.research.ibm.com/hypervisor.

[11] Integrated Virtualization Manager on IBM System p5, Dec. 2006. URL: http://www.redbooks.ibm.com/redpapers/pdfs/redp4061.pdf.

[12] Kernel Based Virtual Machine. URL:http://kvm.qumranet.com/kvmwiki.

[13] T. Jaeger, R. Sailer, and Y. Sreenivasan. Managing the Risk of Covert Information Flows in Virtual Machine Systems. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, France, June 2007.

[14] P. A. Karger. Multi-Level Security Requirements for Hypervisors. *21st Annual Computer Security Applications Conference (ACSAC)*, Dec. 2005.

[15] P. A. Karger, M. E. Zurko, D. W. Bonin, A. H. Mason, and C. E. Kahn. A Retrospective on the VAX VMM Security Kernel. In *IEEE Transaction on Software Engineering*, November 1991.

[16] *Power.org Standard for Power Architecture Platform Requirements (Workstation, Server)*, Version 2.0, 28 August. 2006, Power.org. URL: http://www.power.org/members/developers/specs/PAPR_Version_2.0_28August06.pdf.

[17] N. L. Kelem and R. J. Feiertag. A Separation Model for Virtual Machine Monitors. In *Proc. IEEE Symposium on Security and Privacy*, 1991.

[18] J. M. McCune, T. Jaeger, S. Berger, R. Caceres, and R. Sailer. Shamon: A System for Distributed Mandatory Access Control. *22nd Annual Computer Security Applications Conference (ACSAC)*, Dec. 2006.

[19] R. Meushaw and D. Simard. NetTop-Commercial Technology in High Assurance Applications. *National Security Agency Tech Trend Notes*, Fall 2000.

[20] M. Nguyen and R. Barker. IBM pSeries Hardware Management Console Security White Paper. URL:http://www.ibm.com/servers/eserver/pseries/hardware/whitepapers/hmc_security.pdf.

[21] B. D. Payne, R. Sailer, R. Caceres, Ron Perez, and W. Lee. A Layered Approach to Simplified Access Control in Virtualized Systems. *Operating Systems Review*, Vol. 41, No. 3, July 2007.

[22] R. A. Meyer and L. H. Seawright. A Virtual Machine Time-Sharing System. *IBM Systems Journal*, Vol. 9, No. 3, Sept. 1970.

[23] J. Rushby. Proof of Separability-A verification technique for a class of security kernels. In *Proc. 5th International Symposium on Programming*, vol. 137 of *Lecture Notes in Computer Science*, pp 352-367, Springer-Verlag, 1982.

[24] R. Sailer, T. Jaeger, E. Valdez, R. Caceres, R. Perez, S. Berger, J. Griffin, and L. Van Doorn. Building a MAC-Based Security Architecture for the Xen Opensource Hypervisor. *21st Annual Computer Security Applications Conference (ACSAC)*, Dec. 2005.

[25] J. Whitmore, A Bensoussan, P. Green, D. Hunt, A. Kobziar, and J. Stern. *Design for MULTICS Security Enhancements*, ESD-TR-74-176, Electronic Systems Division, Hanscom AFB, MA, Dec. 1973.

[26] XenSource. URL:http://xenbits.xensource.com/xen-unstable.hg.