RC24154 (W0701-086) January 18, 2007
Computer Science

IBM Research Report

Managing the Risk of Covert Information Flows in Virtual Machine Systems

Trent Jaeger*, Reiner Sailer, Yogesh Sreenivasan*
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

*Penn State University
SIIS Lab
University Park, PA 16802

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home .

Managing the Risk of Covert Information Flows in Virtual Machine Systems

Trent Jaeger
tjaeger@cse.psu.edu
Penn State University, SIIS Lab
University Park, PA 16802

Reiner Sailer
sailer@us.ibm.com
IBM T. J. Watson Research Center
Hawthorne, NY 10532

Yogesh Sreenivasan
sreeniva@cse.psu.edu
Penn State University, SIIS Lab
University Park, PA 16802

ABSTRACT

Flexible mandatory access control (MAC) enforcement is now available for virtual machine systems. For example, the sHype MAC system for the Xen virtual machine monitor is part of the mainline Xen distribution. Such systems offer the isolation of VM systems with the flexible security of MAC enforcement. A problem is that such MAC VM systems will only be assured at modest levels (e.g., Common Criteria EAL4), so they may contain covert channels. Covert channels are often difficult to identify and harder to remove, so we propose an approach to manage possible covert leakage to enable verification of security guarantees. Typically, covert channels are outside of access control policies, but we propose an approach that includes both overt flows and covert flows to assess the possible risk of information leakage due to their combination. We define the concept of a risk flow policy *that describes the authorized risks due to covert flows. In this paper, we evaluate the ability of four policy models to express risk flow policies. Further, we examine how such policies will be enforced in VM systems. We find that variants of the Chinese Wall model and Bell-LaPadula model have features necessary to express risk flow policies. Further, we find that such policies can be enforced in the context of sHype 's Type Enforcement model.*

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Information Flow Controls*

General Terms

Security, Measurement, Management

Keywords

Chinese Wall policy, covert channels, information flow secrecy

1. INTRODUCTION

Virtual machine (VM) systems are gaining in popularity, due to there ability to run multiple, complete systems efficiently on commodity hardware [6, 1]. VM systems provide a virtualized interface to hardware resources that enables sharing of such resources among operating systems that are designed to manage all hardware. Each operating system and its applications run in a single VM that is typically isolated from the other VMs, except through network communications or storage. However, mandatory access control (MAC) systems for VM systems have emerged [26, 22, 23].

MAC enforcement on VM systems controls inter-VM communications, whether they are on a single machine or across machines. Thus, such MAC enforcement is both stronger than traditional VM isolation, as even network communication is controlled, and more flexible than traditional VM isolation, as local VM interactions may now be enabled (i.e., if permitted by the MAC policy). Such MAC VM systems promise comprehensive control of system information flows. However, due to functional requirements (e.g., Xen uses Linux kernel services and drivers to bootstrap function), these emerging systems are unlikely to have a high-level of formal assurance [5]. For example, there is an effort to assure Linux with multilevel security enforcement at EAL4 [11], but it is unlikely that higher assurance will be pursued.

Our challenge is to ensure that MAC enforcement is performed with reasonable risk, despite the lack of complete formal assurance. In particular, we are concerned that MAC controls may be subverted by covert channels [17, 25]. A *covert channel* is a mechanism not explicitly designed for communication that may be used to signal data to another party. For example, if two parties share access to a disk, they may use it as a covert channel by controlling the exhaustion of the disk's storage space. Where overt communication channels are enforced by explicit authorizations, and we have some tools to check comprehensive coverage of authorizations to these channels [12, 35], covert channels are difficult to identify and perhaps impossible to eliminate completely. In this paper, we explore ways to manage the risk in covert channels on VM systems.

The system that we investigate in this paper is the sHype security architecture [26] for the Xen hypervisor [1] VM system. sHype adds authorization hooks to Xen's (overt) communication mechanisms to authorize inter-VM communication on the same machine. It uses a Type Enforcement (TE) model [3] to describe the inter-VM communications that are authorized in a system. TE is expressive enough to specify the expected policies, such as isolation policies and lattice policies [27]. However, a policy in terms of overt communication channels does not enable management of the risk

caused by covert channels.

The risk due to covert channels on a VM system is that any VM on the system may be able to communicate with any other VM through such a channel. For example, when sHype loads a VM, a covert channel may exist that enables that VM to leak information to another VM that may be listening on that channel. The implementation of sHype does not knowingly contain covert channels, but it is not assured not to contain them. sHype enables the expression of a policy to manage such risk: administrators can define conflict sets among VM labels to prevent the concurrent execution of unauthorized VMs. An earlier example can be found in defense environments where data communications are partitioned onto different networks based on its secrecy level. For example, top-secret data is transmitted on a separate network from unclassified data. Although the Bell-LaPadula policy [2] restricts information flow at the access class level, it is impractical to have a network per access class. Therefore, some risk of a vulnerability (i.e., a covert channel is only one vulnerability that is a concern in this network design) is taken by the defense community to minimize its cost.

To express the accepted risk, sHype uses a version of the Chinese Wall policy model [4]. The Chinese Wall policy allows *freedom of choice* in data access, but once a choice is made, future accesses are limited by that choice. In a VM system, the idea is that the system may run VMs of any label, but once a VM is loaded on the system, the choice of subsequent VMs to load is limited. Using the military security labels are an example, if we load a VM labeled *secret*, we may allow other VMs that are *top secret* or *confidential* to run on the same system, but not one that is labeled *unclassified*. This choice permits a risk that data will be leaked from *secret* to *confidential* via a covert channel, but prevents the risk that data will be leaked to *unclassified* VMs. Since the Chinese Wall model permits the freedom of choice envisioned and restricts subsequent accesses based on that choice, it seems like a good candidate model. However, we find that there are problems in using the Chinese Wall model for this purpose, and care must be used to ensure that data is not leaked inadvertently.

In this paper, we study the problem of defining a policy model for expressing the accepted risk of covert data leakage in VM systems. We develop a model of the covert channel leakage problem in VM systems, and identify limitations in the initial sHype approach. Our model identifies a set of information flows that may exist due to a combination of overt and covert information flows. We call these *risk information flows*, so the policy to manage such flows is called a *risk flow policy*. We then explore the application of various access control models in to express risk flow policies, including information flow models, such as Bell-LaPadula [2] and Caernarvon [29], and the Chinese Wall model [4]. We find that risk flow policies can be defined by two types of models: (1) information flow models that express access ranges, such as the Caernarvon model (and available in previous models, such as the SeaView model [21]) and (2) flow constraint models, such as Chinese Wall, although a different expression than traditional conflict sets is necessary. Risk flow policies expressed using information flows are more static, as all the required flows need to be defined in advance, but fail-safe and easier to manage. Risk flow policies expressed using flow constraint models are more flexible and easier to express, but such flexibility may result in challenges, particularly for distributed systems. Since neither approach is monotonically better, further work will be necessary to determine the conditions of use for each.

The paper is structured as follows. In Section 2, we provide some background on the Chinese Wall model, its use in sHype , and the limits of its use. In Section 3, we develop our model of information flow control that accounts for the risks that result from possible covert channels. In Section 4, we evaluate the impact of different policy models on the expression of risk flow policies. In Section 5, we define a system that uses both positive and negative risk flow policies to manage VM information flows. In Section 6, we examine some issues with the approach. In Section 7, we examine related work. Finally, in Section 8 we conclude and outline future work.

2. PROBLEM

In this section, we provide some background on the Chinese Wall security model, and discuss how it is applied in sHype for managing the risk due to potential covert channels. We identify some issues with the current approach that motivates our investigation of the management of covert information flows in this paper.

2.1 Chinese Wall Model Background

Chinese Wall Policy, identified by Brewer and Nash [4], is a hybrid security policy that addresses both confidentiality and integrity. In contrast to Bell-LaPadula Model [2], which is used in Military and Government sectors, Chinese Wall Model aims to describe real-world information flow policies for data owned by commercial and business entities.

The environment of an investment house is the most common example for this model. The main idea is that a single consultant should not have access to information about two corporations that are in competition with one another because such information creates a conflict of interest in consultant's analysis. However, the consultant is free to advise corporations that are not in competition with each other. The U.S. government has passed several laws to improve and formalize this model.

This model categorized the corporate information into three hierarchical levels,

- At the lowest level, we have *objects* of the database which are items of information (files) related to the company.
- At the intermediate level, we have *company dataset* (CD) which contains objects related to a single company.
- At the highest level, we have *conflict of interest* (COI) contains which contains the datasets of companies in competition.

Each object O is associated with a company dataset $CD(O)$ and a conflict of interest class $COI(O)$ to which the company dataset belongs.

The Chinese Wall model defines requirements for reading data (i.e., the *simple-security property*) and for writing data (i.e., the $\star$-security property). An object O can only be read if the subject has accessed a prior object O' belong to the same dataset (i.e., $CD(O) = CD(O')$) or the objects conflict of interest set is new (i.e., $\forall O', COI(O') \neq COI(O)$).

The simple-security emphasizes a *freedom of choice* where a subject may have access to a variety of objects, and only when a choice is made are the restriction implied by that choice enforced.

The requirements for writing are much more restrictive for the Chinese Wall policy. To fulfill the $\star$-property for writing requires that the simple-property be fulfilled for the target object O and that the writer has only read data from that company data set of O (i.e., $\forall O', CD(O') = CD(O)$). The problem is that if a write occurs after a subject has read another company data set, then the two company's data are mixed. For example, if the consultant can write data of company A into an object of company B, then they may be a path to leaking this data to a third company C in conflict with A. Rather than limit the information flows from B to ensure that no flow from A to C is possible, the Chinese Wall model prevents any writing outside of a company. Thus, the consultant is limited to one company's data set of information at a time (i.e., if any writes occur).

2.2 sHype Chinese Wall Model

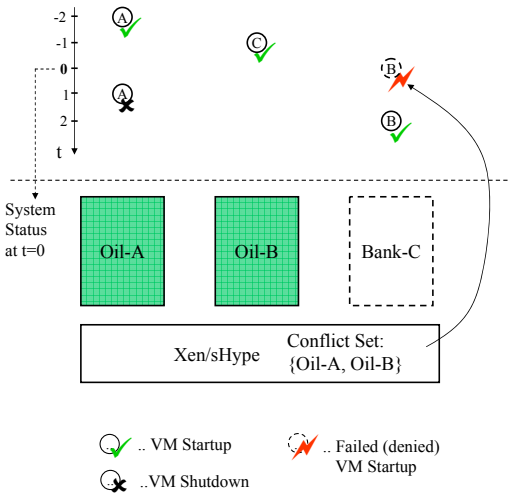

Figure 1: A demonstration of the sHype Chinese Wall model interpretation. Since Oil-A and Oil-B are in a common conflict set, a Oil-B VM may not be loaded on the system until the Oil-A VM is terminated.

Figure 1 shows the sHype interpretation of the Chinese Wall policy. Initially, the Xen hypervisor allows the execution of any VM of any label. However, subsequent load requests will only load the requested VM if it is not in a Chinese Wall conflict set with any currently running VM. We see from Figure 1 that the Oil-B VM is not loaded as long as the Oil-A VM is running, but once the Oil-A VM is terminated the Oil-B VM may be loaded.

The key feature of the Chinese Wall model to sHype is *freedom of choice*. Because the VM systems are largely independent of the VMs that will run on them, we do not want to constrain our use of systems until there is a basis for such constraints. The simple-security property of the Chinese Wall model enables expression of this notion.

The sHype approach assumes a model where only the concurrently executing VMs may communicate covertly. In our example, since the Oil-A VM is no longer running when the Oil-B VM is allowed to run, then there is no way that a covert channel can be used by the Oil-A VM (e.g., by a Trojan horse) to leak information to a waiting process in the Oil-B VM. Other assumptions are possible, however. A more conservative assumption would be that any VM can receive covert information and pass it on. The sHype approach aims to balance functionality, policy simplicity, and security. Here, we study the more conservative assumption to determine what effect this would have on policy simplicity, security, and functionality.

The restrictive $\star$-security property of the Chinese Wall model is not enforced in the sHype interpretation. The Type Enforcement model [3] is used to manage overt channels, so in theory, the TE should prevent unauthorized leaks, although that is not made explicit in the TE policy or its enforcement in sHype .

Consider the three Chinese Wall labels {Bank-C, Oil-B, Oil-A} and a conflict set {Oil-B, Oil-A} from Figure 1. Initially, Bank-C and Oil-A are running simultaneously on the same hypervisor. If the TE policy permits information to flow from Oil-A to Bank-C and Bank-C to Oil-B, then an information flow that violates the Chinese Wall conflict requirement is possible. In the example, the Bank-C VM executes concurrently with both the Oil-A and Oil-B VMs, so it can clearly leak information between the two if it has the TE permissions. Also, the Bank-C VM may write data to a persistent store, so another Bank-C VM may be able to leak the data. While such an information flow can be determined from an analysis of the TE policy [16, 33, 9], we cannot detect the case where potential covert flows may combine overt flows to leak information from the TE policy alone.

2.3 Summary

Our goal is to develop a model to express the requirements on acceptable information flows risks due to covert channels. The sHype model's use of Chinese Wall policies is a start, but its interpretation of possible covert flows does not cover all flows and the information leakage due to a combination of overt and covert flows is not evaluated. In general, we need to describe the acceptable information flows assuming a model of what covert channels are possible. From this, we want to compute restrictions on the execution of VMs on a system based on preventing covert flows that, when combined with overt flows, violate our acceptable information flows.

3. MANAGING COVERT FLOWS

In this section, we develop a model of the risk of information flows caused by potential covert channels. By including covert channels, which have historically been outside the realm of formal access policies, in our flow policy model, we will be able to manage the risk of potential, undesirable information flows in a verifiable way. We first define the concept of a risk flow policy that specifies which information flows may be risked under the possible presence of covert channels. We then develop our model of what constitutes a

covert information flow.

3.1 The Risk Information Flow Model

The goal is to limit the information flows that may be enabled by the presence of a covert channel in a Xen/sHype system. Note that the set of information flows that we allow will be a superset of the information flows allowed via overt channels (i.e., normal access policy).

First, we have the traditional set of information flows.

Definition 1: Overt Information Flows.. *An* overt information flow*, $v \in V$, is an information flow from one subject X to another subject Y that is authorized by the access control policy rules (e.g., Bell-LaPadula). $v : X \rightarrow Y$.*

In the sHype case, the TE policy defines the allowed overt information flows between VMs in a VM system. Other policy models may be used to define information flows, such as a lattice policy [27].

Definition 2: Covert Information Flows.. *A* covert information flow*, $c \in C$, is an information flow from one subject X to another subject Y enabled by a covert channel within a VM system. These are not authorized by the overt information flow policy, but there may be a risk of such potential covert channels. The covert information flows make this risk explicit. $c : X \rightarrow cY$.*

We assume a covert information flow is bidirectional. That is, $(X \rightarrow cY) \Rightarrow (Y \rightarrow cX)$, each covert flow implies a reverse flow. Some covert flows that leak data to external entities, such as side-channels, are not included in the model.

Definition 3: Risk Information Flows.. *The* risk information flows *R are the union of the covert and overt information flows. Such risk flows are* transitive*, like information flows in general, so we say that there exists a risk flow between subjects X and Y if any combination of covert and overt flows can be used to connect X and Y. A risk information flow between X and Y is indicated by a special flow designation. $r : X \rightarrow rY$.*

The risk information flows (or simply *risk flows*) specify all the possible ways that we risk that information may flow in the system. These flows are a combination of those we intend (i.e., via the access control policy) and those that we did not intend (i.e., via covert channels), but may have enabled through our assignment of VMs to VM systems. Since covert channels are included in our model, we can now reason about the impact of the combination of overt flows and the risk of covert flows in a formal manner.

Definition 4: Risk Flow Policy.. *A* risk flow policy*, $p \in P$, consists of a set of statements that specify the risk flows allowed in the VM system. $p : X \rightarrow rY$. We note that transitive flows may be implied by the policy.*

We note that we may use a risk flow policy model that specifies either the flows allowed or those prohibited, as the opposite is implied.

3.2 Computing Covert Flows

On a MAC-enabled VM system (e.g., the sHype system), the trusted computing base (TCB) (e.g., in Xen, the Xen hypervisor and the privileged VM, called dom0) manages access to shared resources and authorizes overt information flows between Xen VMs. In doing this, the TCB may enable a covert channel. Covert channels come in two varieties, storage channels and timing channels, where the former utilizes access to a shared resource (e.g., such as hypervisor structures such as memory maps) and the latter utilizes resources that may be used to "time" operations (e.g., shared processor caches). Storage channels can often be identified, and more importantly, they can be eliminated by proper design (e.g., partitioning resources by access class).

Timing channels, on the other hand, are both more difficult to detect and may be impossible to prevent in practice. Solutions to prevent timing channels include *fuzzy time* [10, 34] where the execution times of operations that may lead to covert channels is varied to distort the channel. Such mechanisms must slow the performance of operations to such a degree that a slow operation under the control of an adversary cannot be distinguished from a slow operation under the control of the system. Such slowdowns are often not acceptable relative to simply buying a separate machine.

As a result, we assume that all VMs on a single machine may communicate using a covert channel supplied (inadvertently) by the TCB. Also, we associate the set of covert information flows with the TCB itself.

Definition 5: TCB Covert Information Flows. *The* TCB Covert Information Flows *are a set of subjects $x \in X$ associated with a VM system's TCB.*

Thus, at any time a TCB is associated with a set of VM labels that may have shared information covertly. Such flows depend on what has happened during a system *run*. We define a system run as a sequence of VM start and VM stop events that indicate both the sequence of VM loads and the concurrent execution of VMs. Thus, given a run of a system, how do we define such a set? There are several considerations to make. First, we assume that the TCB does not itself leak information, but it provides (possible) covert channels that its VMs may use to leak information. The implication of this assumption is that two VMs of different labels must be run for any covert leakage to occur (may be bidirectional). Second, we assume that the TCB completely removes any remnants of a VM when it is terminated. Thus, assumption further requires that the VMs run concurrently, as no leakage from a VM is possible once that VM has terminated. Third, leakage is transitive. If a VM A ran concurrently with VM B at time t, VM A was terminated at time $t+1$, and VM C is started at time $t+2$, then there is a transitive leak of A's data to C via B. Thus, it is important to consider the *overlap* relation between VMs, where VMs A and B are said to overlap if they run concurrently at any time.

Definition 6: TCB Covert Information Flows at Time t.. *At time t, the* TCB Covert Information Flows *are the set of subjects on whose behalf a VM is running at time t, and the transitive closure of the* overlap *relation for those VMs.*

When a new VM of subject X is to be loaded, the overt information flows of the policy and the TCB covert information flows at that time must be evaluated to determine

if the combination of flows results in a violation of the risk flow policy.

4. EVALUATING RISK POLICY MODELS

Next, we examine the effectiveness of expressing and enforcing risk flow policies in a variety of policy models: Chinese Wall [4], Bell-LaPadula [2], Caernarvon [29], and a combination of Chinese Wall and Type Enforcement [3]. We evaluate the models for two tasks: (1) risk policy specification and (2) risk policy enforcement. First, we use the policies to specify the risk flows permitted. Some models specify risk flow policy positively (by flows allowed) and some negatively (by flows denied). We identify that and discuss the impact. Second, we must enforce our risk flow policy in the context of loading a new VM onto a VM system. We examine enforcement in the context of describing overt information flows with these models. We must ensure that the risk flow policy is satisfied by the combination of overt flows authorized and the current state of TCB covert information flows (called covert flows in this section).

4.1 Chinese Wall Policy

We examine how we specify our risk flow policy as a Chinese Wall policy and evaluate the legality of our resulting risk flows using the Chinese Wall semantics. That is, the resulting combination of covert and overt information flows must adhere to simple and $\star$-security property semantics of the Chinese Wall policy.

Using the Chinese Wall policy simple-security property, our risk flow policy specifies a set of unauthorized risk flows. For example, in Figure 1, the risk flow policy is {Oil-A, Oil-B}, which specifies no information flows between Oil-A and Oil-B, in either direction, are allowed. The Chinese Wall conflict of interest set, implies that any flow between any member of the conflict set is prohibited. Like the traditional Chinese Wall policy, the risk flow policy may consist of multiple conflict sets.

We note that the Chinese Wall model is not effective for expressing unidirectional risk flow policies, such as lattice policies. Suppose we want to prevent secret data from leaking to unclassified subjects. The conflict set {secret, unclassified} does not accurately reflect the risk constraint because write-up information flows from unclassified to secret are permitted in a lattice policy (i.e., assuming unclassified is dominated by secret). An alternative Chinese Wall model interpretation is the *aggressive Chinese Wall* model of Lin [20]. In this model, each subject is listed with its conflicts. Thus, we would have a conflict set for secret that is $COI(secret) = \{unclassified\}$, but unclassified has no conflict with secret. In general, a risk flow policy language must be able to express both bidirectional and unidirectional constraints.

In general, the Chinese Wall $\star$-security property semantics are too restrictive for enforcing risk flow policy. In the classical Chinese Wall model, no subject (i.e., a VM) may write data once it has received an information flow from a different subject (i.e., VM with a different label). As shown in Figure 1, the Bank-C VM may enable a transitive risk flow between Oil-A and Oil-B. To prevent such problems, the Chinese Wall model's $\star$-security property would prohibit Bank-C from reading Oil-A's data and writing to either Bank-C or Oil-B. While we must prevent writing to Oil-B, we may not want to prevent writing to Bank-C. Since the Chinese Wall policy does not restrict information flows outside of conflict sets, its *star*-security property interpretation must be excessively restrictive.

This means no data flow between VMs of different labels will be permitted, so we cannot use Chinese Wall to enforce risk flow policy. However, the expression of conflict sets for unauthorized risk flows using an aggressive Chinese Wall policy may still be useful.

4.2 Bell-LaPadula Policy

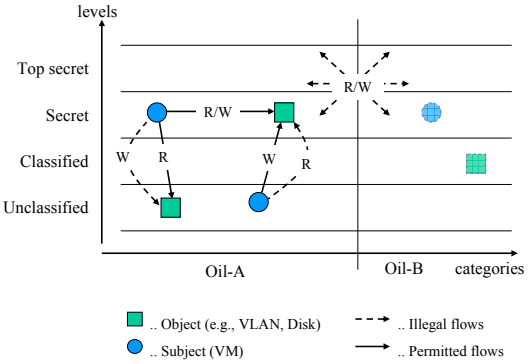

Figure 2: Information flows in the Bell-LaPadula Model. Subjects can write-up and read-down within their category set.

Since we are managing information flows, it makes sense to consider the Bell-LaPadula policy [2]. Bell-LaPadula is a lattice policy that defines what information flows are allowed. Both secrecy levels (e.g., top-secret and unclassified) and category sets (e.g., using Oil-A, Oil-B, etc.) may be used (see Figure 2).

Bell-LaPadula requires unidirectional information flows between different subjects, where any information flow is permitted. For example, a secret subject may only read from a less-secret subject. Writing can only be permitted in the other direction in the lattice.

For specifying risk flow policies, Bell-LaPadula enables expression of two types of risks: (1) leakage between conflicting subjects (i.e., categories) and (2) leakage of higher secrecy information to less secret subjects (i.e., levels). The first type implies no information flow is allowed between the two subjects in either direction. For example, subjects with disjoint category sets may not communicate. The second type implies that data may be leaked in one direction only (i.e., from less secret to more secret). This would be captured by traditional lattice relationships (i.e., dominance enforces flow in one direction).

Covert flows are bidirectional by definition, so category sets achieve the desired semantics (since they restrict in both directions). However, the lattice semantics result in only

being able to specify the most limited risk. Basically, if we consider lattice relationships, only a single label can execute at a time on a VM system that uses Bell-LaPadula policies to express risk flow policy. Any VM of a second label would result in the risk of a covert channel with a bidirectional flow violating the Bell-LaPadula policy. This is similar to the problem with $\star$-security property interpretation in Chinese Wall policies above.

4.3 Caernarvon Policy

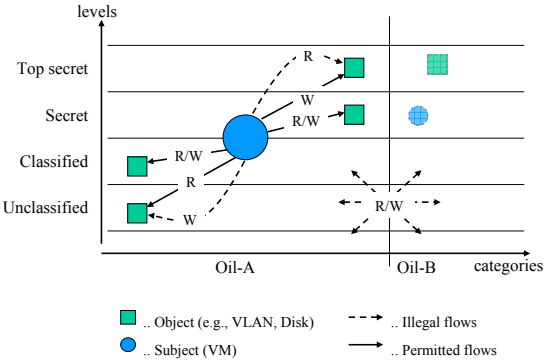

Figure 3: Information flows in the Caernarvon Model. In addition to the traditional read-down and write-up flows of the Bell-LaPadula model, Caernarvon subjects (where trusted) may read/write in a range within the lattice. For example, this subject can read/write secret and classified data.

The Caernarvon policy [29] is a generalization of such lattice policies (see Figure 3). Here, a subject may be able to read and write within a range of labels. Caernarvon secrecy labels define read and write clearances for subjects, where the write clearance defines the lowest clearance that the subject can write and the read clearance defines the highest clearance that the subject can read. Caernarvon permits read-up and write-down within this range.

A Caernarvon risk flow policy is a positive expression of risk flow policy. Because of its lattice heritage, it can directly specify unidirectional flow constraints. We can define a range (e.g., top-secret to secret) where bidirectional flows are permitted, and this range implies a unidirectional flow constraint with subjects outside the range. This implies that lower secrecy subjects may only write-up and higher secrecy subjects can only read-down.

To express a bidirectional flow constraint (e.g., between two company's data), we must define disjoint ranges, one for each member of the conflict set. For example, we would define one Caernarvon subject to access Oil-A data and another to access Oil-B data. There may be other categories of access allowed (e.g., Bank-C), but the ranges must not intersect. Caernarvon supports intersecting ranges, but risk flow policies will have unauthorized leaks if intersecting ranges are used (see Section 6.1 for a more detailed discussion).

Caernarvon supports a lattice model for categories as well, which could be helpful in expressing risk flow policies. In Figure 1, we show that Bank-C may want to communicate overtly with both Oil-B and Oil-A. In Caernarvon, such communications would be upgraded to the category sets {Bank-C+Oil-B} and {Bank-C+Oil-A}. Thus, we would automatically keep the overt information flows isolated, if separate Bank-C VMs were used. Since the same VM is used in the example, the resulting label combines all three which would violate the risk flow policy overtly. If the communications were covert, the covert information flows between Bank-C and Oil-A and Oil-B individually would be added, so this would also violate the risk flow policy.

Caernarvon does not explicitly support the notion of freedom of choice as the Chinese Wall model does. The assumption is that everything has an initial label, which defines its range. In a VM system, any of the labels may be chosen initially (or whenever no VMs are running, assuming secure object reuse). However, once a VM label is chosen, the Caernarvon range is identified. Because the range defines the risk flow policy, Only VMs in this range can be then be executed. Such ranges must be predetermined, and must cover the entire scope of Caernarvon classes (see Section 6.1 for further discussion).

4.4 Type Enforcement Policy

Enforcement [3] (TE) policy to express VM access rights. TE is not an information flow policy, but it is straightforward to convert a TE policy to an information flow policy [14, 15]. Once converted to an information flow policy, the overt information flows are known. The current covert information flows can then be combined to determine the current risk information flows in the system.

As a means for expressing risk flow policies, TE could be used to describe the risk information flows allowed. These could be additional information flows described as covert-only flows whose risk is accepted. Since TE is expressed in terms of specific objects rather than information flows, such a TE policy would be rather low-level.

An alternative is to use TE for enforcement and another policy for expressing risk flows. For example, the current sHype architecture uses only a subset of the classical Chinese Wall model for expressing risk flows, verifying the risk flow policy against the current VM set, rather than the risk information flows as we have defined them here. Further, we note that the Chinese Wall model does not effectively express unidirectional constraints, so the aggressive Chinese Wall model would be necessary to express these.

5. RISK FLOW POLICY APPROACH

In this section, we summarize the analysis above on two points: (1) viable modeling options for expressing risk flow policies and (2) viable enforcement of risk flow policies with traditional, overt policies. On the latter point, it is important that we identify how risk flow policies are enforced accounting for overt information flows as well.

Based on this study, the viable alternatives for expressing risk flow policies appear to be: (1) lattice security models that support a range of bidirectional flows, such as the

Caernarvon policy model, and (2) conflict set models that enable specification of conflicts relative to a particular subject, such as the aggressive Chinese Wall model. Effective models enable isolation between different company data sets (e.g., via conflicts in Chinese Wall and categories in a lattice model) and enable a range of bidirectional communication for permitting risk in lattice secrecy requirements (i.e., risk involves some read-up and write-down in the lattice).

Using the Caernarvon policy model, we would specify a partition of the Caernarvon lattice to describe groups of non-overlapping risk flows. For example, if we require that there are no information flows from Oil-A to Oil-B or vice versa even with the risk of covert channels, then a Caernarvon policy would be specified with each belonging to a separate partition of information flows. Any other risk flow requirements would partition the set of possible information flows further. If only a Caernarvon model is used for risk flow policies, then all the information flows must be partitioned such that no illegal information flows could occur. This could be a difficult task to perform in advance as we discuss in Section 6.2.

Using the aggressive Chinese Wall model, we would specify information flow restrictions for each subject individually. For bidirectional constraints, such as the typical Chinese Wall conflicts, entries are specified for each subject in each conflict set. For example, Oil-A would include Oil-B in its conflict set and vice versa. For unidirectional constraints, such as dominance relationships, only one subject would be assigned the constraint (i.e., the dominating subject to prevent leakage). This representation specifies the flows not allowed, even transitively, by overt or covert flows. In practice, such a policy will also partition the set of possible information flows, eventually, but the partition will be constructed dynamically. Dynamic construction is less labor-intensive for policy specification, but we must avoid preventing a deadlock situation where one application cannot use a necessary information flow that is assigned to a separate partition. We also discuss this further in Section 6.2.

Also, some combination of positive (Caernarvon) and negative (Chinese Wall) risk flow policy expressions is also possible.

For enforcing the risk flow policy, we find that the Caernarvon model or an access matrix MAC model, such as the Type Enforcement model, could be used. The Caernarvon model would naturally be used with a Caernarvon policy. Type Enforcement or any flexible policy model (e.g., that can express an access matrix) could be used with either a Caernarvon risk policy or an aggressive Chinese Wall risk policy or a combination.

Enforcement could work as follows. As described above, we would generate an information flow representation of TE policy. As VMs are loaded, we would extend our VM system's covert flow state (see Section 3.2) to compose a risk information flow graph. At each VM load request, we would verify that the resulting risk information flow graph would not violate the risk flow policy. The risk information flow graph could be constructed efficiently as only covert flows between the new VM and the VM subjects in the covert flow state need to be added. Algorithms to check Caernarvon partition violations would also be efficient (e.g., by marking the subjects with their partitions).

If we use the aggressive Chinese Wall model for risk flow policies, then we have two additional challenges. First, as mentioned above, we need an algorithm to develop the constraints implied by the risk flow policy. Consider Figure 4. In this example, we consider three VM systems where covert flows may be present on each [1]. Starting with the constraint that A and B must not intercommunicate, we need an algorithm that extends our risk flow policy based on the overt and covert information flows that are created dynamically. Second, we need an algorithm to efficiently search the risk information flow graph to test for risk flow policy violations. Basic graph reachability algorithms can be used, but optimizations would seem possible (e.g., via caching).

6. DISCUSSION

In this section, we examine some of the broader issues in implementing a covert channel risk flow policy.

6.1 Intersecting Policies

The risk flow policies discussed in the previous section aim to prevent information flows between conflicting subjects. The Chinese Wall policy defines this through conflict sets that prohibit flows, and the Caernarvon policy defines a range of clearances to which flows are allowed. In our interpretation, both policies will result in a partition of information flows, where the Caernarvon policy would explicitly describe the complete partition, and the Chinese Wall policy would define the partition requirements.

What if information flow sets were allowed to intersect? In general, the Caernarvon policy model can be used to define intersecting information flow sets, and extensions to make the aggressive Chinese Wall policy model also may permit intersecting information flow sets [20]. In Caernarvon, a subject may be defined that can read and write top-secret and secret data, where another can read and write confidential and secret data. These subjects can communicate, because both can read and write secret data, so in this policy top-secret data may be leaked to confidential levels. Using the aggressive Chinese Wall model, the same effect is possible across company datasets. Suppose X conflicts with A and B, but Y only conflicts with B. Therefore, in one run X make leak to Y, but at another time Y may leak to A, violating X's restriction [2].

For a risk flow policy, intersecting ranges are not acceptable. Once one range intersects with another, the risk flows are expanded to include the union of the two sets of flows. Thus, the risk is expanded. For a Caernarvon risk flow policy, the intent would be to describe the range of information flows risked for this subject. If intersections are permitted, then the risk is greater than advertised for this subject. For a Chinese Wall risk policy, intersecting flows would enable risk information flows between conflicting data sets, where the intention is that there be none.

6.2 Partition Construction

Using a Chinese Wall model for risk flow policies, implies a set of flows that should not be allowed. Based on the above discussion, we will ultimately generate a partition of information flows. However, the Chinese Wall policy does

[1] At this point, we ignore covert channels that may span systems.

[2] Unless the Chinese Wall $\star$-security property is enforced, but then no information flows are possible.

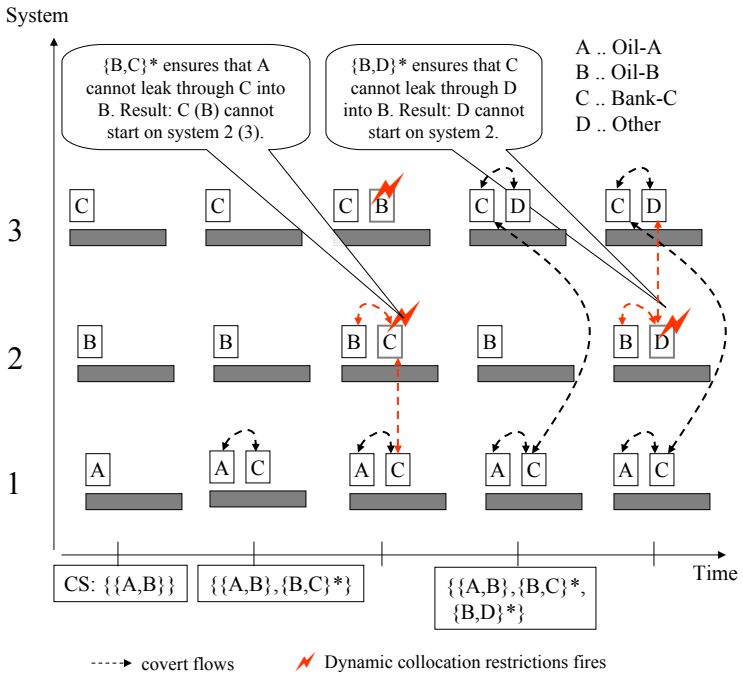

Figure 4: Building Chinese Wall risk flow constraints on the fly. Starting with a risk flow policy conflict between A and B conflicts between B, C, and D are added and enforced dynamically.

not specify how the information flows will be partitioned. Allowing partitions to emerge from dynamic use has the potential of resulting in partitions that do not correspond to application function, resulting in a possible application deadlock. That is, an application may not be able to get access to resources that it would normally have been allowed.

The lattice model (used by Caernarvon and others) handles this problem by using access classes that require access to multiple companies' information. For example, if companies A and B are in conflict, interaction of company C with A would result in information written to an access class requiring permission to both A and C's data. Thus, when C interacts with B the data of A would not be leaked. Since C and A may have interacted with several other subjects, this problem implies the need for some form of *execution monitoring* [30]. The extent to which this should be supported for risk flow policies is a point of future work.

6.3 Distributed Systems Flows

Another problem in risk flow policy management is that information flows may span multiple VM systems. With the emergence of hardware for integrity measurement (e.g., the Trusted Computing Group's Trusted Platform Module [32]), we have proposed an approach to expand reference monitoring across platforms [13, 22], called *Shared Monitoring* or Shamon. Using this approach, the trusted computing bases (TCBs) of multiple VM systems can enforce a single, coherent security policy for a coalition of VMs distributed among the systems. However, the idea permits each system to support the execution of multiple isolated coalitions.

If we account for the risk of covert information flows, then the execution of multiple coalitions on the same VM systems results in some risk information flows. The Shamon approach must account for these risks using a risk flow policy. We envision that Shamon will set the risk flow policy when the overt policy is set. However, some coalitions may already be running on one of the VM systems, which would violate the risk flow policy. This would prevent that VM system from participating in the coalition. As described in the previous section, balancing function and risk must be managed. Distributed enforcement environments, like the Shamon, further emphasize this problem.

7. RELATED WORK

Below, we summarize other related work, in particular, work that investigates the semantics of the Chinese Wall policy and other models that enable dynamic policy choices,

7.1 Chinese Wall Studies

In the original Paper by Brewer and Nash [4], authors argued that although it is possible to represent some aspects of Chinese Wall model using Bell-LaPadula, the Bell-LaPadula model for confidentiality does not emulate Chinese Wall model in its entirety. Therefore, the Chinese Wall model must be regarded as distinct from Bell-LaPadula model. Later, Sandhu [28], demonstrated that the Chinese Wall Model is just another lattice-based model, so it can be represented within Bell-LaPadula framework by making a

proper distinction between users and subjects.

Alves-Foss *et al* argue that it should be possible to cleanup all information related to a particular object and thus tear down part of the Chinese wall with the approval of an external agent [31]. In particular, they argue that conflict-of-interest information may actually conflict only for a limited period of time. Even in sHype we follow a similar approach: we assume that whenever a VM is destroyed or migrated, the VM system clears all of its state.

In other applications of the Chinese Wall model, it has been used as an authorization platform for mobile agents [7]. In this implementation of Chinese Wall model, hosts decide on the authorizations of a mobile agent based on its past behavior. In another approach [18], authors use the Chinese Wall security policy to prevent access to private information in the web. In this approach, organizations are given free access to all items of private information and later access is restricted based on their previous access.

7.2 Execution Monitors

Execution Monitoring (EM) [30, 19, 8] is a class of enforcement mechanisms that monitor the execution steps of a system and terminate the current execution if it violates a security policy. In general, EM monitors the execution traces of the program under consideration and can also be used for debugging, tracing and auditing purposes. Execution Monitoring can be enforced with the help of reference monitors by capturing all security-relevant events and forwarding them for validity checks. A TPM can also be considered as a kind of execution monitoring system.

Schneider defines the execution traces of a system by finite or infinite sequences and security policies as a subset of possible program execute traces [30]. A system satisfies the security policy if and only if its events are a subset of a legal trace. Nagatou and Watanabe argue that typical information flow policies can be enforced by execution monitoring mechanisms, and also such enforcement mechanisms will be able to detect covert channels at run time [24]. Fong claims that to enforce the Chinese wall policy we require only a shallow access history of previously granted accesses [8]. More precisely, to make a decision we only require the set of previously granted accesses not the entire access events or the actual sequencing of access events.

In this work, we do not detect covert channels, but rather, manage the risk of their presence. Further, we find that it is the concurrency of execution that is important, which is a specific interpretation of events.

8. CONCLUSIONS

In this paper, we defined the concept of a *risk flow policy*. A risk flow policy describes the information flows that are permissible given the risk of covert channels. Managing covert channel flows is a problem for emerging medium-assurance VM systems that enforce mandatory access control (MAC), such the Xen sHype system. Customers want assurance that valuable information on such systems is not leaked to competitors, but the systems are not assured against covert leakage. Rather than go to "air gap" systems, we propose managing the risk of such possible information flows using the risk flow policy. In this paper, we investigated the ability of four policy models to express risk flow policies. We found that common models, such as the Chinese Wall model used in the current version of sHype , did not express the covert-channel constraints our model required. We identified two models that enable risk flow policy expression: (1) the Caernarvon model, a lattice model that enables the expression of read-write ranges within a lattice and (2) the aggressive Chinese Wall model, a derivative of the Chinese Wall model that expresses conflicts from the perspective of each subject. These combined with sHype 's Type Enforcement model are capable of specifying and enforcing risk flow policies. A number of challenges remain to provide an effective approach to risk flow enforcement, however, such as: (1) ensuring that risk flow restrictions can be extended dynamically in an efficient manner; (2) that risk flows restrictions can be managed with respect to functional goals; and (3) that risk flow policies are effective for distributed systems.

9. REFERENCES

[1] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the Symposium on Operating Systems Principles (SOSP)*, October 2003.

[2] D. E. Bell and L. J. LaPadula. Secure computer system: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306, Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), L. G. Hanscom Field, Bedford, MA, March 1976.

[3] W. E. Boebert and R. Y. Kain. A practical alternative to hierarchical integrity policies. In *Proceedings of the 8th National Computer Security Conference*, 1985.

[4] D. F. C. Brewer and M. J. Nash. The Chinese Wall Security Policy. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 1989.

[5] Common criteria portal. `http://www.commoncriteriaportal.org/`, 2007.

[6] Scott W. Devine, Edouard Bugnion, and Mendel Rosenblum. Virtualization system including a virtual machine monitor for a computer with a segmented architecture. VMWare, Inc., October 1998. US Patent No. 6397242.

[7] Pedro Dias, Carlos Ribeiro, and Paulo Ferreira. Enforcing history-based security policies in mobile agent systems. In *POLICY*, pages 231–234, 2003.

[8] P. W. L. Fong. Access control by tracking shallow execution history. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, pages 43–55, 2004.

[9] A. L. Herzog, J. D. Guttman, D. R. Harris, J. D. Ramsdell, A. E. Segall, and B. T. Sniffen. Policy analysis and generation work at MITRE. In *Proceedings of the first Annual Security-enhanced Linux Symposium*, March 2005.

[10] W-M. Hu. Reducing timing charmers with fuzzy time. In *Proc. of the 1991 IEEE Symposium on Security and Privacy.*, pages 8–20, 1991.

[11] atsec and IBM to make Red Hat Linux a government certified trusted operating system. `http://lwn.net/Articles/156140/`, 2005.

[12] T. Jaeger, A. Edwards, and X. Zhang. Consistency analysis of authorization hook placement in the Linux security modules framework. *ACM Transactions on*

Information and System Security (TISSEC), 7(2):175–205, May 2004.

[13] T. Jaeger, P. McDaniel, L. St. Clair, R. Cáceres, and R. Sailer. Shame on trust in distributed systems. In *Proceedings of the* 1^{st} *Workshop on Hot Topics in Security*, July 2006.

[14] T. Jaeger, R. Sailer, and X. Zhang. Analyzing integrity protection in the SELinux example policy. In *Proceedings of the 12th USENIX Security Symposium*, pages 59–74, August 2003.

[15] T. Jaeger, R. Sailer, and X. Zhang. Resolving constraint conflicts. In *Proceedings of the 2004 ACM Symposium on Access Control Models and Technologies*, June 2004.

[16] Trent Jaeger, Xiaolan Zhang, and Antony Edwards. Policy management using access control spaces. *ACM Transactions on Information and System Security*, 6(3):327–364, 2003.

[17] B. W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

[18] Frans A. Lategan and Martin S. Olivier. A chinese wall approach to privacy policies for the web. In *COMPSAC*, pages 940–944, 2002.

[19] Jay Ligatti, Lujo Bauer, and David Walker. Edit automata: Enforcement mechanisms for run-time security policies. *International Journal of Information Security*, 4(1–2):2–16, February 2005. (Published online 26 Oct 2004.).

[20] T. Y. Lin. Chinese wall security policy-an aggressive model. In *Proceedings Fifth Annual Computer Security Applications Conference*, pages 282–289, Tucson, AZ, 1989.

[21] T. F. Lunt, D. E. Denning, R. R. Schell, M. Heckman, and W. R. Shockley. The SeaView security model. *IEEE Transactions on Software Engineering*, 16(6):593–607, 1990.

[22] J. McCune, S. Berger, R. Cáceres, T. Jaeger, and R. Sailer. Shamon: A system for distributed mandatory access control. In *Proceedings of the 22st Annual Computer Security Applications Conference*, December 2006.

[23] Robert Meushaw and Donald Simard. Nettop: Commercial technology in high assurance applications. Available at: `http://www.vmware.com/pdf/TechTrendNotes.pdf`, 2000.

[24] N. Nagatou and T. Watanabe. Run-time detection of covert channels. In *ARES*, pages 577–584, 2006.

[25] N. E. Proctor and P. G. Neumann. Architectural implications of covert channels. In *Proceedings of the Fifteenth National Computer Security Conference*, pages 28–43, October 1992.

[26] Reiner Sailer, Trent Jaeger, Enriquillo Valdez, Ramón Cáceres, Ronald Perez, Stefan Berger, John Griffin, and Leendert van Doorn. Building a MAC-based security architecture for the Xen opensource hypervisor. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005)*, Miami, FL, USA, December 2005.

[27] R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, 1993.

[28] Ravi S. Sandhu. Lattice-based enforcement of chinese walls. *Computers & Security*, 11(8):753–763, 1992.

[29] Gerhard Schellhorn, Wolfgang Reif, Axel Schairer, Paul A. Karger, Vernon Austel, and David Toll. Verification of a formal security model for multiapplicative smart cards. In *Proceedings of the 2000 European Symposium on Research in Computer Security*, pages 17–36, 2000.

[30] F. B. Schneider. Enforceable security policies. *ACM Transactions on Information and Systems Security*, 3(1):30–50, 2000.

[31] J. Sobel and A. Alves-Foss. A trace-based model of the Chinese Wall security policy. In *Proceeding of the 1999 National Information System Security Conference*, 1999.

[32] Trusted Computing Group. `http://www.trustedcomputinggroup.org/`, March 2005.

[33] Tresys technology, SETools policy tools for SELinux. `http://www.tresys.com/selinux/selinux\_policy\_tools.shtml`.

[34] J. T. Trostle. On modelings a fuzzy time system. In *Proc. of the 1993 IEEE Symposium on Security and Privacy.*, pages 82–89, 1993.

[35] X. Zhang, A. Edwards, and T. Jaeger. Using CQUAL for static analysis of authorization hook placement. In *Proceedings of the 11th USENIX Security Symposium*, pages 33–48, San Francico, CA, USA, August 2002.