A Practical Chinese Wall Security Model in Cloud Computing

Tien-Hao Tsai*, Yen-Chung Chen*†, Hsiu-Chuan Huang*†, Pei-Ming Huang* and Kuo-Sen Chou*
*Information & Communication Security Lab
Chunghwa Telecom Laboratories
Taoyuan, R.O.C.
{p1t1r,yzchen,pattyh,peiming,cksp}@cht.com.tw
†Institute of Computer Science and Engineering
National Chiao Tung University
Hsinchu, R.O.C.

***Abstract*—Virtualization technology is widely adopted in clouds to meet the requirements of rapid provision and on-demand scalability in cloud computing. Although virtualization improves the usage of hardware devices and flexibility, it brings new security challenges. Users face a new type of attacks, called inter-VM attack, which targets at the VMs running on the same physical machine. To eliminate the possible inter-VM attacks from competitors, we propose a centralized control mechanism based on the Chinese Wall security policy to forbid deploying and running the competitors' VMs on the same physical machines so that physical isolation is achieved. We build the Chinese Wall Central Management System (CWCMS) with the proposed centralized control mechanism in an internal-built experimental cloud. CWCMS effectively manages the VMs and enforce the Chinese Wall security policy in the cloud. Furthermore, CWCMS employs the graph coloring algorithm to achieve the better utilization of cloud resources.**

Keywords- Chinese Wall security policy; Virtualization; Cloud

I. INTRODUCTION

Virtualization technology plays a very important role in the construction of clouds. Virtualization allows organizations to use the hardware devices more efficiently and more flexibly. With virtualization, the hardware is simulated that allows multiple virtual machines (VMs) to run concurrently on a single computer. Each VM runs its own operating system (OS) and applications on top of the virtual hardware, accessing its own resources as if it run on an independent computer. Through the control of a virtual machine monitor (VMM), also called hypervisor, VMs share the same hardware resources but do not interfere with each other.

Although virtualization improves the usage of hardware devices and flexibility, it brings new security challenges. Users face a new type of attacks, called inter-VM attack, which targets at the VMs running on the same physical machine. In a virtual environment, a VM is likely attacked not only by external computers but also by other VMs that reside on the same physical machine. A VM can attack another VM directly, or attack the hypervisor first and then control the hypervisor to attack other VMs on the same machine. The systems of competitors may run on the same machines if they rent VMs from the same cloud vendor. Successful inter-VM attacks launched by competitors will be more harmful than ever. To eliminate the possible inter-VM attacks from competitors, we propose a centralized control mechanism based on the Chinese Wall security policy (CW policy) [1] to forbid deploying and running the competitors' VMs on the same machines in the clouds. Given the conflict of interest sets, the centralized control mechanism deploys VMs with conflict of interests on different physical machines so that physical isolation can be achieved. Our work enhances the security of cloud computing to protect the systems and data in the clouds.

We build the Chinese Wall Central Management System (CWCMS) with the proposed mechanism in an internal-built experimental cloud that uses Kernel-based Virtual Machine (KVM) [2] as the solution of the virtualization platforms. CWCMS manages and coordinates the deployments of VMs on virtualization platforms, ensuring the deployments follow the CW policy specified. The deployments and migrations of VMs are restricted by the conflict of interest relations when the Chinese Wall security model is adopted. If the conflict of interest relations are not set reasonably or the VMs are not deployed in a well-planned way, the cloud vendors must allocate more physical machines to deploy VMs. Instead of deploying one VM on one available physical machine arbitrarily and testing whether a conflict of interest is found on that machine, CWCMS employs the graph coloring algorithm to analyze the conflict of interest relations among VMs and find a good solution to distribute VMs to different physical machines. It makes the management of the CW policy much easier and more effective.

II. BACKGROUND

The CW policy is originally a commercial security policy, which is adopted to separate people with conflict of interests and prevent them from accessing and sharing information so that investment decisions will not be affected. In 1989, Brewer and Nash proposed a mathematical theory which implemented the CW policy and introduced it into computer security. Initially, a person (a subject) has freedom to access data (an object) in any dataset. Once he accesses a dataset, a Chinese Wall is created for him so that he cannot access other datasets

which are in the same conflict of interest class. The CW policy has been extended and applied in various domains [3]-[7].

To protect the hypervisor and the systems on the VMs, resource isolation and privilege management are performed. Each VM is controlled by the hypervisor to access its own resources only, such as memory addresses and disk spaces. And the use of the CPU privileged instructions is limited in VMs to prevent VMs from getting the supervisor permission of the platform. However, the vulnerabilities of the virtualization platforms are still found [8]-[11]. In the virtual environment, we enforce the CW policy to eliminate the possible inter-VM attacks from competitors. Once a VM is loaded and executed on a physical machine, other VMs with conflict of interests are not allowed to work on the same machine. Based on the CW policy, VMs with conflict of interests are separated to run on different physical machines to achieve physical isolation.

III. Related Work

IBM's sHype [12] provides an Access Control Module (ACM) on Xen. It implements the Type Enforcement policy to control the sharing of resources among VMs on a system, and the CW policy to ensure VMs with conflict of interests not to run concurrently on the same system. For the CW policy, it checks the conflict of interest relations when a VM is loaded on a system. If any conflict is found, it refuses to load the VM on that system. Since sHype does not check the conflict of interest relations among VMs in advance, it may try to deploy the VM on different systems several times until one that does not have the conflict is found.

sVirt [13] is a Mandatory Access Control (MAC) security module based on SELinux. Unlike our work which focuses on physical isolation for competitors' systems, sVirt focuses on resource isolation on virtualization platforms. sVirt also prevents bugs in the hypervisor from being used by VMs to attack the host OS or other VMs. Our work eliminates the possible inter-VM attacks from competitors, and sVirt reduces the harm caused by a success attack. Thus, combining our work with sVirt will harden the security of virtualization platforms.

IV. Chinese Wall Central Management System

A. System Architecture

CWCMS is built to provide the centralized control mechanism based on the CW policy. To enforce the CW policy and manage the deployments of VMs effectively and efficiently, a two-layer management model is adopted as shown in Fig. 1. Chinese Wall Central Server in Layer 1 manages the virtualization platforms in the cloud to follow the CW policy. It mainly has three functionalities: (1) management of VMs' conflict of interest sets, and CW policy generation; (2) CW Policy dispatch; (3) VM deployments based on the CW policy. Layer 2 consists of virtualization platforms with Chinese Wall Control Agents installed. Chinese Wall Control Agent is responsible for receiving and enforcing the CW policy in each virtualization platform. Each virtualization platform manages its VMs through the hypervisor APIs. Chinese Wall Central Server in Layer 1 remotely controls the virtualization platforms in Layer 2 by the installed agents.

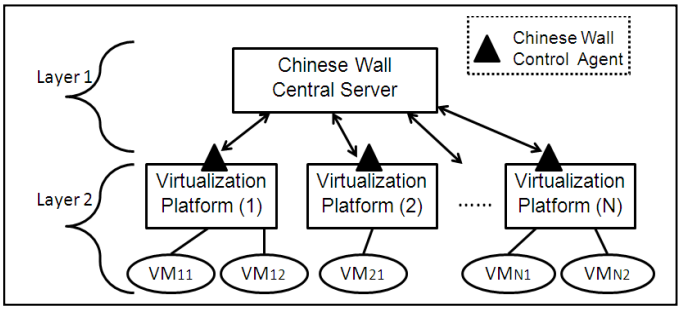

Figure 1. System Architecture of Chinese Wall Central Management System (CWCMS).

VMs are created in accordance with the commands from Chinese Wall Central Server. Chinese Wall Control Agent not only obeys the commands from Chinese Wall Central Server but also monitors the creation, suspension and shutdown of VMs on each machine and informs Chinese Wall Central Server of the changes. In order to enforce the CW policy, hooking hypervisor APIs and intercepting the abnormal VM creation in virtualization platforms are also performed.

B. Conflict Relations and Graph Theory

CWCMS adopts graph theory to analyze the conflict of interest relations among VMs and decides how to deploy VMs that not only follows the CW policy but also achieves good utilization of cloud resources. If a CW policy specifies that VM A and VM B have a conflict of interest relation, the machine VM A runs on would forbid VM B to run on it, and vice versa. The exclusive relations could be drawn as an undirected graph, in which each node represents a VM and each edge represents a conflict relation between two VMs. Assume VM A, B and C have the conflict relations: A<->B, B<->C, C<->A, the graph could be shown as Fig. 2(a). The VMs with conflict relations should be deployed to different machines based on the CW policy. In other words, two VMs cannot be deployed to the same machine if their corresponding nodes are connected by an edge. We apply the graph coloring algorithm to color each node where two neighbors would not apply the same color as shown in Fig. 2(b). The graph coloring algorithm ensures correct deployments of VMs that follow the CW policy.

We apply the graph coloring algorithm from Michael A. Trick [14,15] and make some adjustments to meet the requirements for VM deployments. Trick's algorithm finds a minimum coloring that uses as few colors as possible for a graph. This algorithm, as shown in Fig. 3, assigns colors to the nodes of the maximum clique and then to the remaining nodes, finding a better coloring by changing nodes' colors (excepting the nodes in the maximum clique). This algorithm can solve complex coloring problems in a quick and effective way.

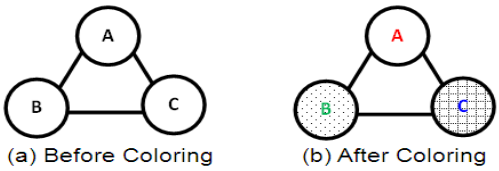

Figure 2. An representation of conflict relations in an undirected graph.

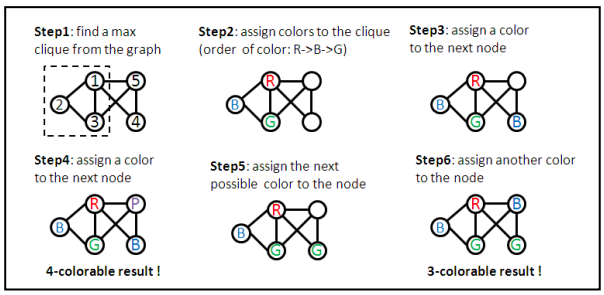

Figure 3. An example of Michael A. Trick's graph coloring algorithm[14].

One issue is about the resource limitation. If nodes are assigned to the same color, the corresponding VMs could be deployed to the same machine. However, it is impossible to deploy all the VMs with the same color into a single machine in practice due to the hardware resource limitation. The maximum number of VMs that can be deployed on a machine without performance downgrade, denoted by *LimitPerMachine*, is measured in advance. The other issue is that using a minimum coloring does not guarantee the best utilization of cloud resources. We revise Trick's algorithm to set a restriction on the maximum number of nodes for each color, which is a multiple of *LimitPerMachine* and denoted by *LimitPerColor*. Another issue is that, to reduce the number of colors, it may be better to change the colors for the colored nodes when a new node is joined. In the cloud, changing colors means moving the VM from one physical machine to the other. According to [16], migration brings seconds of downtime, and the overheads of multiple VM migration are much higher than single VM migration. We suggest the migration would be done during the VM off-peak time.

V. SYSTEM ANALYSIS

In our experiment, the graph instances in [17] are used to simulate the conflict relations among the VMs. Table I shows a short description of them. The limit *LimitPerMachine* depends on the hardware capability and the virtualization solution adopted. In our KVM experiment cloud, each physical machine contains 4*Intel X5660(6-cores) and 96GB RAM. Each machine allows 10 VMs to run on it without a significant performance downgrade. Thus, *LimitPerMachine* is set to 10. Different deployment models influence the usage of machines and flexibility. Four deployment models are developed:

(a) Deploying VMs without CW policy: This model provides the basis for comparison. In this model, the CW policy is not applied and VMs are deployed in a machine as many as possible to minimize the amount of machines in use. The amount of machines in this model indicates a lower bound of machines allocated to meet the resource requirements.

(b) Deploying VMs with Trick's algorithm: The CW policy is applied and VMs are grouped by color using Trick's algorithm. VMs with the same color are deployed to a set of machines, where each only contains VMs with the same color.

(c) Concentrating VMs by colors: Unlike Trick's algorithm which finds a minimum coloring, VMs are assigned to the first applicable color or a new color if none is qualified.

TABLE I. A SHORT DESCRIPTION OF THE GRAPH INSTANCES USED.

Graph instance name	Number of nodes	Number of edges
(1) queen8_8.col	64	728
(2) queen9_9.col	81	2112
(3) queen10_10.col	100	2940
(4) queen8_12.col	96	1368
(5) jean.col	80	254
(6) myciel5.col	47	236
(7) myciel7.col	191	2360
(8) le450_15b.col	450	8169
(9) le450_5a.col	450	5714
(10) le450_5d.col	450	9757
(11) anna.col	138	493

(d) Concentrating VMs by machines: Unlike Trick's algorithm, non-exclusive VMs are put into a machine as many as possible to minimize the amount of machines in use.

Fig. 4 shows the numbers of required machines for each deployment models. The x-axis represents the graph instances, and the y-axis represents the amount of machines required for each graph instance and each deployment model. The leftmost bar in each graph instance indicates the minimum number of machines that should be allocated for the graph instance.

Comparing Model (b) and (c) which apply the color-grouping deployment strategy, Model (b) that adopts Trick's algorithm is better than Model (c) in most cases. Comparing Model (b) and (d), Model (b) uses less machines when the number of VMs is small. The reason is that Model (b) takes the conflict relations of all the VMs into consideration to determine how to deploy VMs. When the number of VMs is large, Model (b) uses more machines because there are some 'unstuffed' machines. A machine is 'unstuffed' if the number of VMs that run on it is less than *LimitPerMachine*. Although Model (d) uses fewer machines in the later case, it is not flexible for VM migration. Applying Model (d) often leads to a full status for most machines. When the CW policy is enabled and no extra machines are provided, a VM with high loading cannot easily find a candidate machine to immigrate. Contrarily, if we group VMs by color like Model (b), VMs with the same color could exchange arbitrarily.

In terms of the number of physical machines, the concentrated strategy achieves better utilization. For VM movement or exchange, the color-grouping strategy is more flexible. When analyzing the VM deployments on each machine by applying Model (b), we find some drawbacks if Trick's algorithm is adopted directly in VM deployments. The goal of Trick's algorithm is to find a minimum coloring so it would search all colors in use and find the first applicable color when choosing the color. The drawbacks include: (i) Solitary assignment: The number of nodes for each color may not be a multiple of *LimitPerMachine*. Some machines may have only few VMs that run on them. It would be waste if there are too many 'unstuffed' machines. (ii) Inclined assignment: Too many VMs are assigned to the former-used colors and only some nodes are assigned to the later colors.

We revise Trick's algorithm to adopt the concentrated strategy whenever it needs to choose a color. The revised algorithm sets a restriction on the maximum number of nodes for each color, *LimitPerColor*.

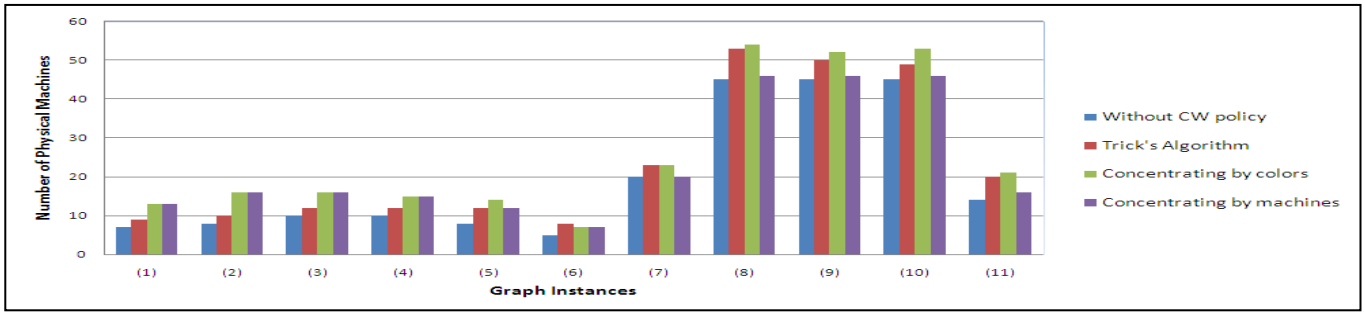

Figure 4. Number of physical machines required for different deployment models. The graph instances described in Table I are used.

TABLE II. THE COMPARISON OF NUMBER OF INCREASED MACHINES FOR APPLYING DIFFERENT ALGORITHMS AND LIMITS.

Algorithm	**Increased physical machines for applying the CW policy**	
	Average of increased machines	**Average of increased machine ratio**
Trick's algorithm	3.73	0.272
Revised Trick's algorithm with *LimitPerColor* set to 10	1.27	0.132
Revised Trick's algorithm with *LimitPerColor* set to 20	2	0.226
Revised Trick's algorithm with *LimitPerColor* set to 30	2.63	0.248
Revised Trick's algorithm with *LimitPerColor* set to 40	3.45	0.273
Concentrating VMs by machines limited by 10	3.27	0.37

Table II shows the comparison for average numbers of increased physical machines for all graph instances in Table I when applying different algorithms and limitations. The average of increased machines is the number of increased physical machines divided by the number of physical machines required when the CW policy is not applied. Table II shows the benefit of using the revised Trick's algorithm. If *LimitPerColor* is set too large, it is more likely to result in inclined assignment. Taking both the number of machines and the convenience for movement into account, the revised Trick's algorithm with *LimitPerColor* set to three times *LimitPerMachine* is recommended.

VI. CONCLUSION

In this paper, we propose a centralized control mechanism based on the Chinese Wall security policy to forbid deploying and running the competitors' VMs on the same physical machines so that physical isolation is achieved. We present CWCMS to manage and deploy conflict VMs to proper different physical machines. We also tailor Trick's graph coloring algorithm for VM deployments, solving management problems caused by complex CW policies and achieving better utilization of cloud resources.

REFERENCES

[1] David D. C. Brewer and Michael J. Nash, "The Chinese Wall security policy," in Proceedings of the IEEE Symposium on Security and Privacy, 1989, pp. 215-228.

[2] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. "kvm: the Linux virtual machine monitor," in OLS '07: The 2007 Ottawa Linux Symposium, July 2007, pp. 225--230.

[3] T. Y. Lin, "Chinese Wall security policy-an aggressive model," Proceedings of the Fifth Annual Computer Security Application Conference, December 1989, pp. 286-293.

[4] P.C.K.Hung and Guang-Sha Qiu, "Implementing conflict of interest assertions for web services matchmaking process," in Proceedings of the IEEE International Conference on E-Commerce (CEC), 2003, pp. 373-380.

[5] T. Y. Lin, "Managing information flows on discretionary access control models," IEEE International Conference on Systems, Man, and Cybernetics , 2006, vol. 6, pp. 4759-4762.

[6] F. A. Lategan and M. S. Olivier, "A Chinese Wall approach to privacy policies for the web," in Proceedings of the 26 th Annual International Computer Software and Applications Conference (COMPSAC'02), 2002, pp. 940-944.

[7] Yu-Cheng Hsiao and Gwan-Hwan Hwang, "Implementing the Chinese Wall security model in workflow management systems," 2010 International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2010, pp. 574-581.

[8] " CVE-2009-3525, " 2009. [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3525

[9] " CVE-2009-1758, " 2009. [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-1758

[10] " CVE-2008-4993, " 2009. [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4993

[11] " CVE-2008-3687, " 2008. [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-3687

[12] R. Sailer, T. Jaeger, E. Valdez, R. C´aceres, R. Perez, S. Berger, J. L. Griffin, and L. van Doorn. "Building a MAC-based security architecture for the Xen opensource hypervisor," in Proceedings of the Annual Computer Security Applications Conference (ACSAC), December 2005.

[13] "sVirt," [Online]. Available: http://selinuxproject.org/page/SVirt

[14] D. S. Johnson and M. A. Trick, "Cliques, coloring, and satisfiability: Second dimacs implementation challenge," in Dimacs Series in Discrete Math. and Theoret. Comput. Sci. 36, 1996

[15] A. Mehrotra and M. A. Trick, "A column generation approach for graph coloring," INFORMS Journal on Computing, vol. 8, no. 4, 1996.

[16] W. Voorsluys, J. Broberg, S. Venugopal and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in CloudCom '09: Proceedings of the 1st International Conference on Cloud Computing, Berlin, Heidelberg. Springer-Verlag, 2009, pp. 254–265.

[17] "Graph Coloring Instances," [On-line] . Available: http://mat.gsia.cmu.edu/COLOR/instances.html