Project Report

Virtual Machine Security

Submitted in partial fullfilment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

by

Sachin Boban
B090009CS

Under the guidance of

Dr. Priya Chandran

Department of Computer Science & Engineering
National Institute of Technology, Calicut
Kerala - 673601

April 2013

Department of Computer Science & Engineering

National Institute of Technology Calicut

Certificate

This is to certify that the project work entitled "**Virtual Machine Security**", submitted by Sachin Boban (B090009CS) to National Institute of Technology Calicut towards partial fulfillment of the requirements for the award of Degree Of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the work carried out by him under my supervision and guidance.

Project Guide
Dr Priya Chandran
Professor

Place : Calicut
Date :

Head of Department

Office Seal

Acknowledgement

A project is a culmination of hardwork and efforts made by many people, directly and indirectly, by their valuable support and encouragement. I would like to express my sincere gratitude to everyone who helped me in completing this project.

I take this opportunity to express my profound gratitude and deep regards to my guide Dr. Priya Chandran, Professor, Department of Computer Science & Engineering, National Institute of Technology Calicut, for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her time to time shall carry me a long way in the journey of life on which I am about to embark.

I also take this opportunity to express a deep sense of gratitude to Dr. S D Madhu Kumar, Associate Professor and Head, Department of Computer Science & Engineering, National Insttute of Technology Calicut for providing all the facilities required during this project.

I also thank Mrs. Anu Mary Chacko, Assisstant Professor, Department of Computer Science & Engineering, National Institute of Technology, Calicut who was our Faculty Advisor and Project Coordinator for all her love, support and encouragement without which this project would have never been possible.

I also thank all my friends who have been of great support. A special mention to Thridev Suvarnan, Sreenath S Kamath, Yasser Murikadan Prasad Vasudevan and Rahul P, who had always been there for me with support and encouragement when I needed the most.

Lastly, I thank almighty, my parents, brother and sister, for their constant encouragement, love, care and support without which, this project would not be possible.

Sachin Boban

Problem Statement

A study on the existing security issues associated with the rollback mechanism in Virtual Machines and exploration of possible solutions.

Abstract

Virtual Machine Monitors(VMM) is now a hot topic in both academia and industry. VMMs emulate an existing architecture, and facilitate multiple instances of independent virtual machines and resulting in an efficient use of computing resources, in terms of both energy consumption and cost. Hardware virtualization is a core issue in cloud computing. However, virtual computing platforms cannot be deployed securely by a simple drop into existing systems. As virtual machines replace the existing real machines, they give rise to radically different and more dynamic user models. However this undermine security architecture of many organizations. Further, VMMs provide many useful mechanisms like rollback, which can result in unpredictable and harmful interactions with the existing security mechanisms in use. The focus of this project is on the security issues associated with rollback of a virtual machine. It aims at finding a solution for these issues.

Contents

1 Introduction **3**
- 1.1 History . . . 3
- 1.2 Virtual Machine Architecture . . . 3
- 1.3 Security Issues . . . 4

2 Checkpoint & Rollback-Recovery **5**
- 2.1 Checkpoint . . . 5
- 2.2 Rollback-Recovery . . . 6

3 Security Issues **7**
- 3.1 Invalid Randomness . . . 7
- 3.2 Invalid OTP System . . . 7
- 3.3 VM Rollback Attack . . . 8

4 VM Rollback Attack **10**
- 4.1 Reason for the vulnerability . . . 10
- 4.2 Simple Solutions and Feasibility . . . 10
- 4.3 Proposed Solution . . . 10

5 Solution Ideas **12**
- 5.1 Disable Rollback . . . 12
- 5.2 User Authentication . . . 12
- 5.3 Using Nested Hyervisor . . . 12
  - 5.3.1 Approach . . . 12
  - 5.3.2 Drawback . . . 13
- 5.4 Rollback Sensitive Data Memory . . . 13
  - 5.4.1 Approach . . . 13
  - 5.4.2 Drawbacks . . . 14

6 Architectural Solution **15**
- 6.1 Approach . . . 15
  - 6.1.1 Role of Application Process . . . 15
- 6.2 RSDM-A Controller . . . 16
  - 6.2.1 RSDM Access Key . . . 17

6.2.2 Ownership and RSDM Table 17
6.3 Interfaces 17
6.3.1 VM Creation 17
6.3.2 VM Migration 18
6.3.3 Accessing RSDM 19
6.4 Limitations 19

7 Summary & Future Scope **20**
7.1 Summary 20
7.2 Future Scope 21

References **22**

Chapter 1

Introduction

1.1 History

A Virtual Machine Monitor (VMM) provides a layer of software abstraction between the operating system(s) and hardware of a machine to create the illusion of one or more Virtual Machines (VMs) on a single real machine[1]. VMMs were introduced in early 1960s as a software-abstraction layer that partition existing hardware into two or more virtual machines. This gave a compelling way to multiplex the then expensive mainframe hardware. But in 1980s, the introduction of multiplexing Operating System(OS) and cheaper hardware eroded the importance of VMM. However, 2005 saw the revival of VMMs as a hot topic in both academia and industry. As system administrators resorted to running one application per machine, hardware requirements shot up, imposing both cost and management overheads[2]. Rather than a vehicle for multitasking, VMMs are now a solution for security and reliability. Both migration and security are difficult to attain in modern operating systems, than in VMM.

1.2 Virtual Machine Architecture

Virtualization can be done at the process level (as in Operating Systems) or at the system level. In this project the concentration is on the System Virtual Machines. The fundamental idea behind a virtual machine is to abstract the hardware of a single computer (the CPU, memory, disk drivers etc) into several execution environments, thereby creating the illusion that each seperate execution environment is running its own private computer[6]. In any system VM, the VMM provides platform replication. The VMM divides the hardware resources among the different guest operating systems. The guest OS and its applications are under the hidden control of the VMM.[7]. System VMs can be set up either as a classic VM or as a hosted VM (Fig 1.1). In classic VM, the VMM runs with the highest priviledged mode, while

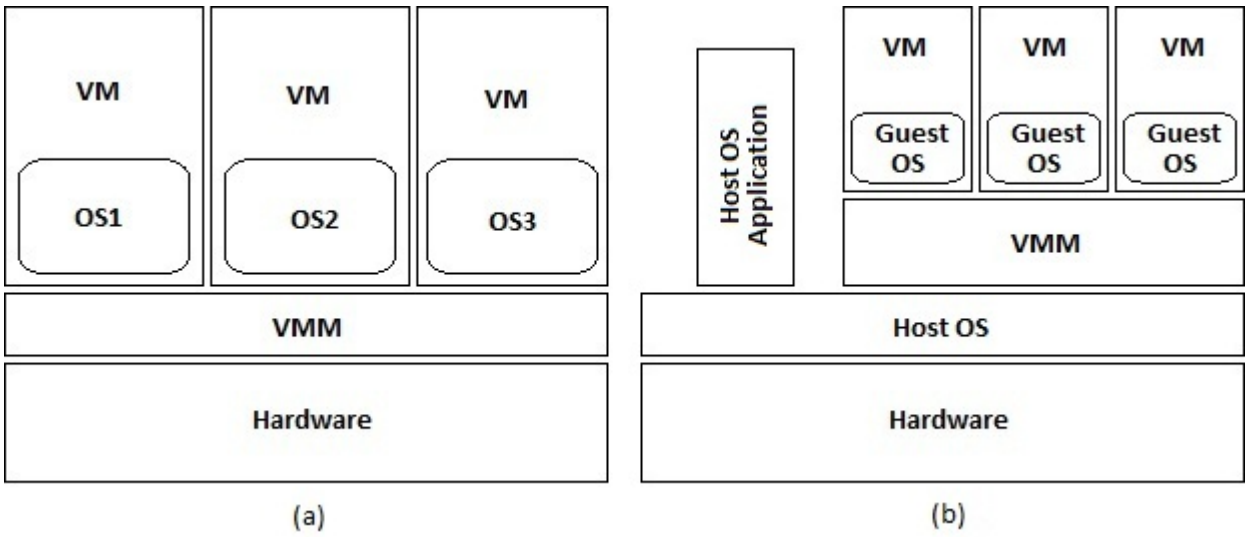

Figure 1.1: (a)Classic VM (b) Hosted VM

all other guest systems run with reduced priviledge level. This is done so that the VMM can intercept all guest istructions that normally would access or manipulate critical hardware resources. In hosted VM, the virtualizing software itself is built over the an existing OS called the Host OS. This approach has the advantage that, the virtualizing software or VMM can rely on the Host OS to provide device drivers and other lower-level services, rather than the VMM[7].

1.3 Security Issues

VMMs provides many functionalities that can be done with all the ease of manipulating a file. These include, creating, copying, sharing, migration, checkpointing and rollbacking of VMs. As the virtual machines replace the existing real machines, they give rise to radically different and more dynamic user models. However this undermine security architecture of many organizations. Further, VMMs provide many userful mechanisms like rollback, which can result in unpredictable and harmful interactions with the existing security mechanisms in use. The focus of this project is the security issues associated with the rollbacking mechanism of virtual machines. The attempt is to find a solution to these security issues posed by rollbacking.

Chapter 2 will give a brief idea about creating a snapshot and rollback-recovery. Chapter 3 discusses about the various security issues. Chapter 4 contains a brief discussion on VM rollback attack. Solution ideas are disscussed in chapter 5. Future plans are mentioned in chapter 6.

Chapter 2

Checkpoint & Rollback-Recovery

Rolbacking a VM essentially involves two important sub processes. At first we need to save the stable state of a VM i.e. create a snapshot. The second would be to restore the VM to the already created snapshot.

2.1 Checkpoint

Checkpoint involves saving the states of all ongoing process in a VM. This includes saving CPU states, memory, disks etc. At a particular time, there can be a number of ongoing processes within a single guest OS. Its required to save the state of each of these processes. A set of checkpoints, one per each process in the system form a checkpoint for the entire system. However such a checkpoint must be consistent. Rollbacking from a set of inconsistent checkpoints may cause many problems. Consider the following situation. Suppose there are two processes P1 and P2 as shown below. Both P1 and P2 interact with each other by message passing. Suppose at a time, T1, P1 sends a message M to P2, and P2 receives it at time T2. Suppose that the snapshot was created at a time T' between T1 and T2. As per the snapshot so created P1 says it has already send the required message M to P2. However according to P2 it received no message from P1. Rollbacking on such a snapshot would create a lot of issues. Suppose if P2 was waiting for the message M to start a particular work. This means P2 will be waiting indefnitely thinking P1 is yet to send the message M. So while creating a snapshot, care must be taken so the the snapshot is consistant for each of processes running inside the corressponding VM.

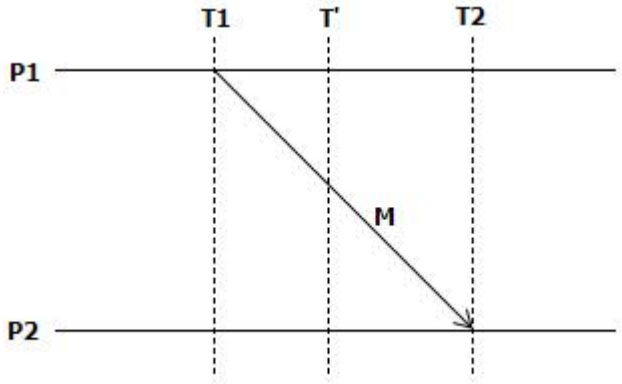

Figure 2.1: Inconsistent Snapshot

2.2 Rollback-Recovery

Rollback-recovery from a consistent set of checkpoints appears deceptively simple. Consider the straight forward recovery method in which, when a process rolls back to checkpoint, it notifies all other processes to rollback to their corressponding checkpoints. However such an approach can result in *livelocks*, where the failure can cause an infinite number of rollbacks, stalling the VM execution[4]. Care must be taken so that rollback-recovery method employed do not create any livelocks.

Chapter 3

Security Issues

Rollback involves restoring the virtual machine to an old check point. However such a restoration results in a lot of security issues. Rollback may expose some patch vulnerabilities, reactivate vulnerable services, re-enable previously disabled accounts or passwords, use previously retired encryption keys, and change firewalls to expose vulnerablities. To put in simple words, while VMs may be rolled back, an attacker's memory of what has been already been sent cannot[2].

3.1 Invalid Randomness

Majority of the cryptographic protocols depend on nonces or session keys. The security of such protocols depend on the freshness of their random number sources[1]. When a VM is rolled back after a random number has been chosen and used, there is a possibility that the random number generated after rollbacking is the same. Such a situation compromises the freshness of the random number chosen. For example, encrypting multiple messages with the same key in stream ciphers will expose the XOR of messages.

In addition to cryptographic protocols, non-cryptographic protocols like TCP that rely on freshness are also at risk. In TCP reuse of initial sequence numbers can allow hijacking attacks. Another example is Zero Knowledge Protocols (ZKP) like Fiat-Shamir authentication, where private key of the user will be compromised if the same nonce is used. Similarly signature systems derived from the ZKPK protocols like Digital Signature Standard (DSS) would compromise the secret signature key if same random number is used to generate two signatures[1].

3.2 Invalid OTP System

Consider the famous one-time password (OTP) system S/KEY. It works as follows. The step begins with a secret key w either provided by the user or

generated by a computer. A secure hash function H is applied on w n times, producing a hash chain of passwords. Only the password generated at last say Kh is stored in the host. When the client consumes the password Kc host makes sure H(Kc)=Kh. The host will then replace Kh by Kc. While the virtual machine rolllbacks, the old Kh will be restored in the rollback of the whole virtual memory and virtual disk[8].

3.3 VM Rollback Attack

In a VMM, the hypervisor is able to suspend a VM at any point of execution, make a snapshot of it and resume later whenever it wishes, without the guest VM's awareness[3]. This process helps in recovering from any VM failure and maintanence. However this can be used by malicious attackers to launch a VM rollback attack. In such an attack, a compromised hypervisor is made to run from an old snapshot without the awareness of the owner of the VM. This results in the loss of a part of VM's execution history. Now such a loss is exploited by the attacker. Security checks can be bypassed

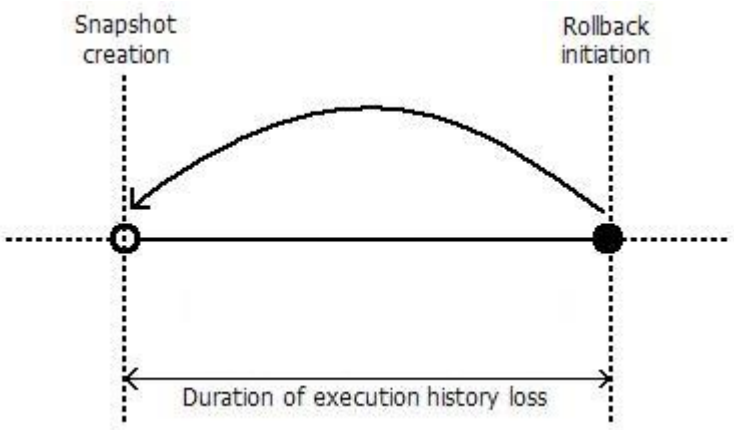

Figure 3.1: Partial Execution Histroy Loss

or critical security updates can be undone by the attacker. Consider the example, where an attacker attempts a brute-force login . Its common for any Operating System to have a restriction on the number of continuos failed attempts during login. Suppose a particular system blocks login for a while after three continuos failed trials. Now the attacker tries a brute-force method to crack the password. After three continuos failed attempts, the attacker can infinitely rollback the VM to an initial state, untill he/she is able to login. The difficulty in preventing such an attack is that its usually hard to distinguish between a rollback attack from a normal suspend/resume

operation. The proposed solution in [3] gives a good insight into the usage of nested hypervisor to solve security vulnerabilities. A brief discussion of the same is made in the next chapter.

Chapter 4

VM Rollback Attack

In a rollback attack, a compromised hypervisor runs a VM from an old snapshot wihtout the user's awareness[3]. When a virtual machine is rolled back to a previous checkpoint, there is a partial loss of execution history since the checkpoint creation. Through such a rollback the attacker can bypass some security checks or even undo some security updates.

4.1 Reason for the vulnerability

Although the Guest OS has security checks to prevent attack such as the one mentioned in Section 3.3, they simply fail in a virtual machine environment. This is because, snapshot is legal and internally consistent, and the rollback attack can penetrate these security checks. Rollback attack is possible primarily because the VMM cannot distinguish between a malacious rollback and a genuine one.

4.2 Simple Solutions and Feasibility

Some prior works simply disables the rollback mechanism as such to ensure security[10]. However this renders the normal operations like snapshot and VM migration unavailable. Moreover, snapshot is required for the virtual machine to recover from any failure. Since the user can easily identify a malacious rollback, another possible solution is to ask the user to authenticate each suspend and resume operation of the virtual machine. However, this would be impractical and annoying to the user.

4.3 Proposed Solution

The proposed solution is based on CloudVisor[5], where user can detect malicious rollbacks using the secure logging mechanism they propose. For

this, rollbacking to the middle of a running epoch is forbidden and the start and end of every eopch are securely logged. It also involves the addition of four additional hypercalls in CloudVisor: *vmboot, vmshutdown, vmsuspend* and *vmresume*. The working of these hypercalls are as follows. *vmsuspend*

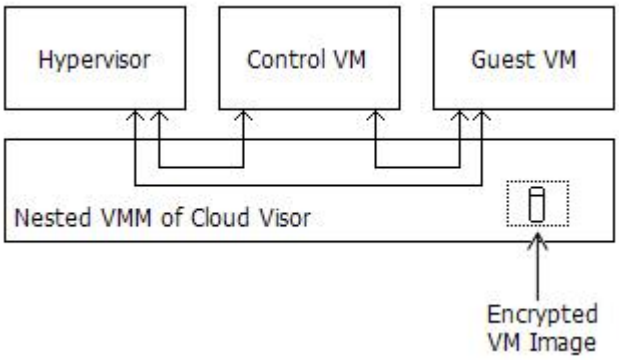

Figure 4.1: CloudVisor Archiecture[3]

and *vmshutdown* are required to make snapshots of the CPU Registers, Memory and Disk image. The CloudVisor would then hash them. The hash is then encrypted with the VM key. This hash is then used as the version for identification [3]. Both *vmresume* and *vmboot* require snapshot and snapshot hash. Hash is also checked and validated. All these four operations are logged into the secure log. This mechanism however is not 100% secure.

Once a cloud operator make a rollback, the user can only consider it as a suspicious action. It is the cloud service provider's responsibility to prove the rollback is genuine, by showing other evidence to explain why it is necessary to do so[3]. The proposed work in [3], however lacks clarity and there is no mention of a verification of the proposed solution based on simulations or theoretical assertion.

Chapter 5

Solution Ideas

The aim of the project is to come up with a solution for the security issues discussed earlier in the report.

5.1 Disable Rollback

The first solution that would come into mind for all these issues is to disable the rollback feature as such. However such a drastic decision would render some normal operations such as snapshot, VM Migration etc unavailable. Consider a cloud environment without rollbacking where a guest OS fails. The user would then have to ask the cloud service provider to recover the failed guest OS, without any loss. So disabling snapshot and rollback is not a viable option.

5.2 User Authentication

Another option to prevent rollback attack is to ask the user to authorize each and every boot/resume, suspend/shutdown, and migration. However this would result in seeking premission from the user each time such an operation is to be performed. This would be annoying and impractical.The user will also have to be provided with proper justification for the particular operation to be done. This would require the users to have some idea about the VM implementational details.

5.3 Using Nested Hyervisor

5.3.1 Approach

Using the approach mentioned in [3], nested hypervisor can be used. As mentioned in chapter 4, majority of the issues are associated with the freshness of the random number and nonce generators. Issue here is that these

generators are also rolled back. The idea is to prevent these generators from rolled back. For this whenever a VM is resumed, as a first step, the states of these generators are saved into the nested hypervisor. After rollback-recovery is completed, the states of the generators are fetched back from the nested hypervisor and restored. In this approach the freshness of these random number and nonce generators are preserved.

5.3.2 Drawback

This approach, however has a scalability issue. Since most of the applications in the VM have these generators of their own, such an approach will require all of them to be saved into the nested hypervisor. Considering the large number of VMs on a single real machine and the applications, each with it's own generators, such an approach is too complex. Moreover while restoring the states, each of the generators must be reset with their own states. Given the huge number, such a mapping will be a huge burden on the VM and the nested hypervisor.

5.4 Rollback Sensitive Data Memory

The drawback of the previous approach was the lack of scalability . The VMM cannot map the freshness sensitive data or states back to the correct source. Inorder to overcome this scalability burden, this new approach requires effort from the VMM and the Guest OS.

5.4.1 Approach

For simplicty, the freshness sensivtive data and generator states is refered as sensitive data. Inorder to ensure secure the freshness, the sensitive data is saved in a special memory within VMM. This is not as trivial as the approach before.

At first VMM will set apart a special memory, called *Rollback Sensitive Data Memory (RSDM),* to save the sensitive data. VMM provides each Guest OS with a RSDM. The Guest OS uses its own RSDM to save the rollback senstive data. The sensitive data can also be updated whenever required. However no Guest OS is allowed a direct entry into its RSDM. Any read or write request to the RSDM is channeled throught the VMM. This restriction ensures that any attempt to access the RSDM content is validated by the VMM. To access its RSDM, the Guest OS makes use of two special instructions: *rsdm_read* and *rsdm_write*. These requests when issued, notifies the VMM.

So far this approach is the same as the one discussed previously, except for the concept of specialised memory, the RSDM. VMM only provides RSDM to the Guest OS. Its the Guest OS that has to manage the data

within the RSDM using *rsdm_read* and *rsdm_write*. When a checkpoint is created, the VMM does not include RSDM in snapshot. After the rollback recovery is completed, the sensitive data remains fresh since the RSDM is left untouched. Remapping of the sensitive data is no longer required.

When a VM migration is required, along with the VM snapshot, the RSDM content must also be transfered to the new device. Note that the Guest OS uses logical address to address the RSDM memory. The VMM will generate the physical address of the RSDM location. So all that the VMM needs to keep track is a base address and a bound value for each RSDM. On the new VM platform, when a newly migrated VM needs to be resumed, only few additional work is required. A new RSDM is to be created and initialised with the RSDM content provided along with the snapshot. This also includes base and bound values of the RSDM. The Guest OS is not bothered by these changes during suspend/resume since it uses logical address.

There arises a question, what if the hypervisor or the VMM itself is malicious? The concept of nested Hypervisor as in Cloud Visor[5], can be applied to ensure RSDM is secure even if the hypervisor is compromised.

5.4.2 Drawbacks

Although the RSDM approach is simple, it has serious drawbacks. First, the Guest OS requires considerable source code editing to implement such a change(i.e. its now a para-virtualization). So Guest OS is aware of the system virtualization. Moreover, this approach rely on the Guest OS to identify the rollback sensitive data. This is possible in case of data or states that belong to the Guest OS itself (like the number of failed log in attempts). However its difficult or sometimes even impossible for the Guest OS to identify the data from the applications as rollback sensitive or not, unless its explicitly specified by the application. This is a serious drawback.

Chapter 6

Architectural Solution

From Section 5.1 one can understand the need for solving rollback issues without disabling rollback-recovery and checkpointing feature in a VMM. In this project the aim is to find a solution for the issues discussed. The idea mentioned in Section 5.3 has scalability issue. RSDM approach mentioned in Section 5.4 solves the scalability part, but requires Guest OS intervention to identify the sensitive data.

Its clear from the drawbacks of RSDM approach (Section 5.4.2) that Guest OS cannot identify the rollback senstive data from the applications. Also, the OS handling of the application data is minimal and is reduced to the handling of *read()* and *write()* system calls. These drawbacks shows that an architecture level solution is requred to handle the rollback security concerns. Moreover, an architecture based solution is considered to be more secure than a software solution.

6.1 Approach

This solution is based on [9] which tries to secure virtualization under a vulnerable hypervisor using architectural support. A similar approach is used, where RSDM itself is implemented with some changes and additional architectural support. Again RSDM is used to store sensitive data. Earlier the hypervisor provides each VM with the RSDM. But in this new approach, it is the architecture that provides each VM with the required RSDM. This new approach with the architecture support is called as *RSDM-A(RSDM with Architecture assistance)*(Figure 6.1). The entry into the RSDM is now under the control of the RSDM-A contoller, which is a part of the hardware.

6.1.1 Role of Application Process

From Section 5.4.2, its clear that Guest OS cannot identify sensitive data, but the owning application can. So in RSDM-A approach, its the application

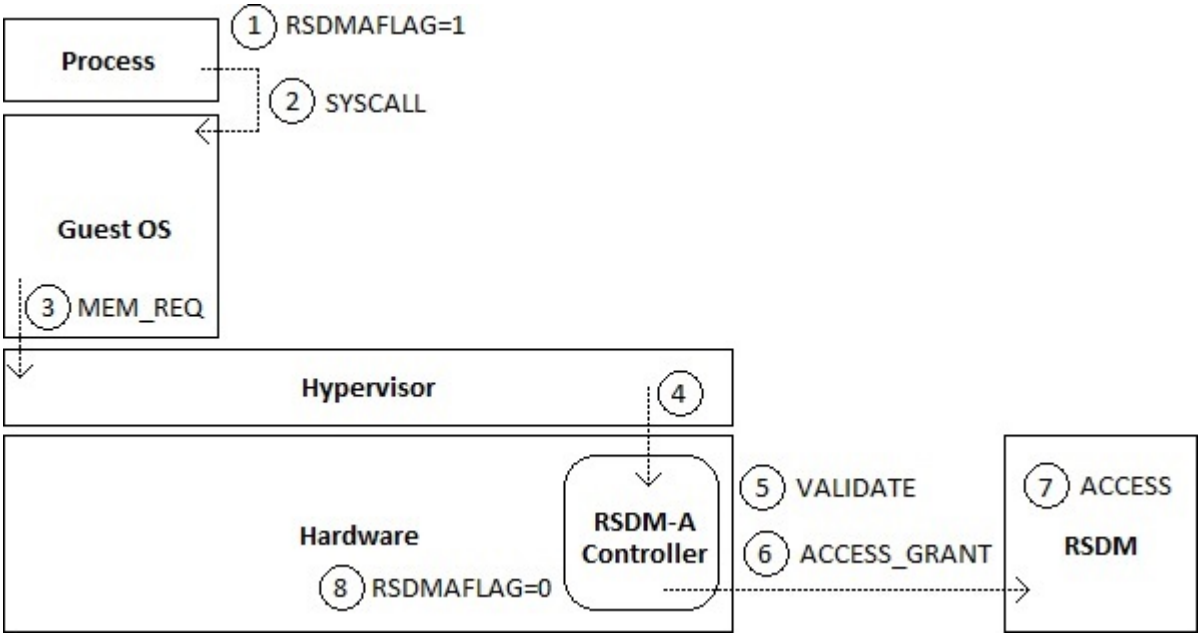

Figure 6.1: RSDM-A Architecture

process which will notify the RSDM mechanism that the incoming data is sensitive. For this application uses a *RSDMAFLAG* (RSDM Access Flag), provided by the underlying architecture. Whenever an RSDM manipulation is required, the application sets the RSDMAFLAG to 1.

Conider the case of *read()* employed to get some RSDM content. Before *read()* system call is invoked, the application process must set the RSDMAFLAG to 1. When the actual memory read occurs, RSDMAFLAG notifies the RSDM-A controller that the read access is from the RSDM-A. Note that inorder to validate the identity of a VM the H-SVM (Hardware-assisted Secure Virtual Machine) mechanism proposed in [9] makes use of a key generated by the H-SVM when the VM was actually created. Similarly inorder to ensure that the entry into RSDM is done only by the coressponding Guest OS which owns the RSDM, VMID must also be supplied to the RSDM-A architecture. Similarly any other entry into the RSDM also follows the same procedure.

6.2 RSDM-A Controller

Architectural support for RSDM does not end with the RSDMAFLAG. In RSDM-A the entry into RSDM is controlled as well as validated by a *RSDM-A controller*, which is provided by the hardware. Inorder to validate the RSDM access request, the RSDM-A controller can make use of a key unique

to each VM. Now All the functionalities of the hypervisor for implementing RSDM is performed by the RSDM-A controller. These functionalities include controlling access to RSDM, alloting RSDM to a new VM, isolation of different RSDMs, keeping track of the physical locations of each RSDM, its ownership etc.

6.2.1 RSDM Access Key

Inoder to verfy that RSDM is accessed only by its owner, each RSDM access must be validated by the RSDM-A controller. For this the RSDM-A controller makes use of the RSDM Access Key. This key is unique to each Guest OS, so that any illegal access by any non-owning Guest OS or a malicious hypervisor can be prevented. For this some key exchange method is to be used between RSDM-A controller and the new VM. Whever a RSDM access is requested, the RSDM-A controller must also be supplied by the corressponding application. The RSDM Access Key can be obtained by the application from the Guest OS or the Guest OS can provide it to the RSDM-A controller during a RSDM request. The later is prefered inoder to keep Guest OS changes to a minimum. Whenever a RSDM memory access request reacher the RSDM-A controller, it checks RSDM Access Key to ensure that the RSDM access is legal. If the key is valid, then the RSDM-A controller grants access.

6.2.2 Ownership and RSDM Table

RSDM is secure only if all the access to the RSDM is channeled through the RSDM-A controller, which such accesses. For this, the phyical memory is tagged with its owner. If the owner is RSDM-A controller, access is granted only to the RSDM-A controller. Inoder to keep track of each RSDM's physical location, the RSDM-A controller maintains a RSDM Table.

6.3 Interfaces

This section explains the functionalities of the RSDM-A controller during various events.

6.3.1 VM Creation

When a new VM added to a machine, at first the Guest OS and the RSDM-A controller must agree on the RSDM Access Key unique to that Guest OS. This key is used by the RSDM-A controller to identify the VM as well as validate the RSDM access. The key exchange mechanism must me done in such a way so that neither other VMs not the hypervisor will get the RSDM Access Key. Once the key is agreed upon, the RSDM-A controller

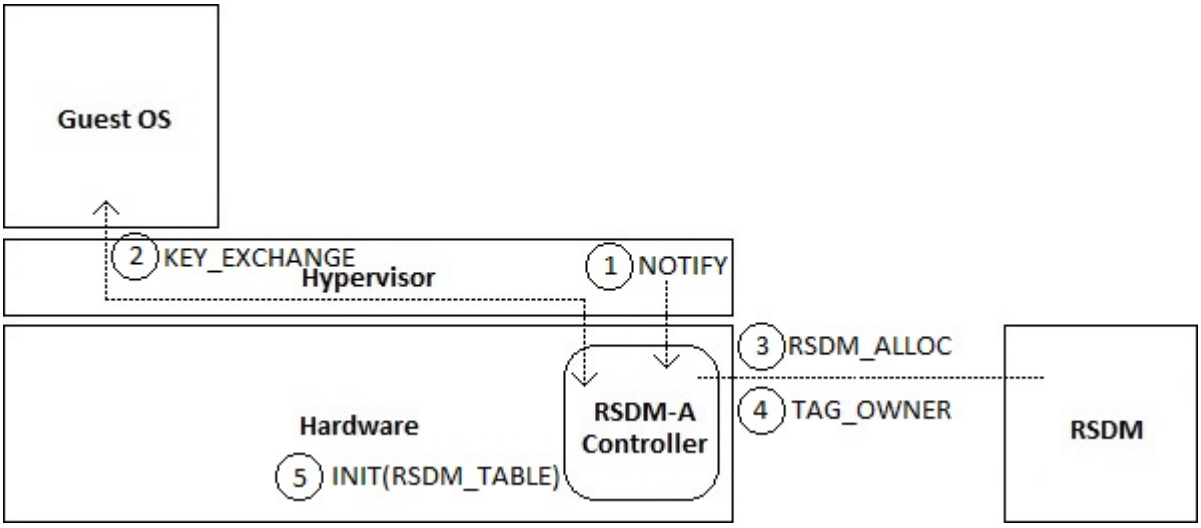

Figure 6.2: VM Creation

allocates RSDM to the newly created VM, tags itself as the owner of the physical locations allocated to the RSDM, and adds these locations to the corressponding RSDM table entry.

6.3.2 VM Migration

As mentioned in Section 5.4, the snapshot of a VM consists of two parts, RSDM content and the non-senstive part saved when the checkpoint is created. Snapshot is required in two cases, suspend/resume and migrating VM to a different physical machine. When VM undergoes a suspend/resume, RSDM remains fresh within the machine. So RSDM will not have much involvement. Now when a VM is migrated, the checkpoint must also be transfered to the new machine as to resume the VM successfully. For this along with the non-sensitive checkpoint data, RSDM content must also be transfered to the new machine. When the VM arrives at the new machine, the RSDM-A controller allocates RSDM to the VM initianlises the RSDM table and tags the physical locations' ownership. Now instead of using a new RSDM Access Key, the old key as such can be used by the Guest OS. This key must be however securely send to the RSDM-A controller of the new machine so that it can do the controlling and access validation. If the key is however already in use, then the machine can create a new key and send it to the new VM. Since the Guest OS uses logical addressing for RSDM, migration will not affect the addressing mechanism.

6.3.3 Accessing RSDM

The application processes or Guest OS manipulates the RSDM using file realted system calls. Before these system calls are made, the requesting process must set the RSDMAFLAG to 1. This will notify the RSDM-A controller that the memory requests are for the RSDM. RSDM-A controller then validates the access by checking the ownership of the RSDM. Once validated, the RSDM-A controller allows the corressponding action to be performed. Once the request is completed, the RSDM-A controller clears the RSDMAFLAG i.e. its set to 0. The same is explained using Figure 6.1

6.4 Limitations

RSDM-A approach relies on the application processes to identify its sensitive data, and to notify the RSDM-A Controller of the sensitive data. So the processes that are sensitive to rollbacking must be altered so that it can make use of RSDM-A architecture to keep the rollback sensitive data fresh. Howvever, one of the major advantage of using virtualization is the support for legacy softwares. Modifying such legacy softwares will be an issue.

Chapter 7

Summary & Future Scope

7.1 Summary

Checkpointing and rollback-recovery are important features of Virtual Machines. These mechanisms help in suspending an resuming VMs, VM Migration, recovering from software failures etc. These mechanisms however can result in unpredictable and harmful interactions with the existing security mechanisms in use. Since rollbacking restores the state of a VM to old checkpoint, there is a complete loss of execution history. State based random number generators gets reset to an old state, so the random number genrator will generate the same random number used early. Thus the random number generator losses its randomness. Cryptographic protocols that requires these random number generators also fails due to the loss of the randomness.

Disabling checkpoint and rollback-recovery mechnisms will render many useful features unavailable. This necessitated the need for finding a solution for the security issues assocaited with the rollbacking mechanisms. RSDM meachnism was proposed where the rollback senstive data is protected by the hypervisor, but the Guest OS still has control over the sensitive data. A special memory called RSDM is allocated to each VM where it can save the rollback sensitive data. Although the Guest OS can manipulate the data within its RSDM, the access is granted only through the hypervisor. This mechanism relied on the Guest OS to notify what to store inside its RSDM.

However the assumption that the Guest OS can identify incoming data as rollback senstive or not does not hold. This necessitated the need for an architecture based solution. RSDM-A mechanism was thus proposed as an architecture solution to solve the security concerns at hand. RSDM-A is an extension of RSDM mechanism where the control of RSDM is given to an RSDM-A Controller inside the hardware. Instead of the Guest OS, the application processes notifies the RSDM-A meachnism whether the incoming data is rollback senstive or not. This simplifies the Guest OS to a very large

extend, keeping the changes made to the Guest OS to a minimum.

7.2 Future Scope

This project proposes a solution with architecture support to solve the security concerns related to rollbacking mechanism in Virtual Machines. Althought a solution is proposed and its validity is theoretically argued, its never validated using implementation or simulation. Moverover the performance of the mechanism is not formally evaluated. This can be done by making use of simulations. The future work can be the verification of the RSDM-A mechanism and its performance evaluation with the help of simulations.

References

[1] Garfinkel, T. and Rosenblum, M. *When Virtual is Harder than Real: Security Challenges in Virtual Machine based Computing Environments.* In HOTOS-X, Santa Fe, New Mexico, June 2005.

[2] Rosenblum, M. and Garfinkel, T. *Virtual machine monitors: current technology and future trends.* Computer, vol.38, no.5, pp. 39-47, May 2005.

[3] Yubin Xia, Yutao Liu, Haibo Chen and Binyu Zang. *Defending against VM Rollback Attack.* Dependable Systems and Networks Workshops (DSN-W), 2012 IEEE/IFIP 42nd International Conference on , vol., no., pp.1-5, 25-28 June 2012.

[4] Koo, R. and Toueg, S. *Checkpointing and Rollback-Recovery for Distributed Systems* IEEE Transactions on Software Engineering, Vol, SE-13, No. 1, January, 1987.

[5] F. Zhang, J. Chen and B. Zang *CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization.* In Proceedings of SOSP, 2011, pp. 203-216.

[6] A. Silberschartz, P.B. Galvin and G. Gagne. *Operating System Principles.* Seventh Edition, John Wiley & Sons Inc. 2006.

[7] Smith, J.E.and Ravi Nair. *The Architecture of Virtual Machines.* Computer, vol.38, no.5, pp. 32-38, May 2005.

[8] Lei Yu, Chuliang Weng, Minglu Li, and Yuan Luo. *Security Challenges on the Clone, Snapshot, Migration and Rollback of Xen Based Computing Environments.* Fifth Annual ChinaGrid Conference (ChinaGrid), 2010. pp.223-227, 16-18 July 2010.

[9] Seongwook Jin, Jeongseob Ahn, Sanghoon Cha and Jaehyuk Huh. *Architectural Support for Secure Virtualization under a Vulnerable Hypervisor.* MICRO-44'11. Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, 2011, pages 272-283.

[10] J. Szefer and R. B. Lee. *Architectural Support for Hypervisor-Secure Virtualization.* ASPLOS 2012. Proceedings of the 17th Annual International Conference on Architectural Support for Programming Languages and Operating Systems, London, 2011.

[11] Smith, J.E. and Ravi Nair. *Virtual Machines.* Elsevier Inc, 2005.