Out of band Systems Management in enterprise Computing Environment

Divyanshu Verma
Dell India R&D Centre.
Divyanshu_Verma@Dell.com

Srinivas G Gowda
Dell India R&D Centre.
Srinivas_G_Gowda@Dell.com

Ashokan Vellimalai
Dell India R&D Centre.
Ashokan_Vellimalai@Dell.com

Surya Prabhakar
Dell India R&D Centre.
Surya_Prabhakar@Dell.com

Technical Reviewers: {Ramkrishna_Rama, Jagadeesh_Raju, Neti_Prasad}@Dell.com

Abstract

Out of band systems management provides an innovative mechanism to keep the digital ecosystem inside data centers in shape even when the parent system goes down. This is an upcoming trend where monitoring and safeguarding of servers is offloaded to another embedded system which is most likely an embedded Linux implementation.

In today's context, where virtualized servers/workloads are the most prevalent compute nodes inside a data center, it is important to evaluate systems management and associated challenges in that perspective. This paper explains how to leverage Out Of Band systems management infrastructure in virtualized environment.

1 Introduction

Out Of Band systems management is the de-facto capability in enterprise computing world to manage physical servers inside a data center. It provides remote administrators the ability to connect, gather server information and at the same time control the servers even in non-OS environment.

Today's enterprise computing environment is dominated by virtualization technology which allows one single server to be used by many virtual machines. In this paper we look at how we can make Out Of Band System management utility scale up to this new challenge of virtualization. We talk about the ways in which an existing systems management utility can be used to handle virtual machines. Later on we also discuss some of the security challenges that may be posed while trying to implement this method.

1.1 Acronyms and Abbreviations

OOB - Out Of Band Management

VMS - Virtual Management Software

VM - Virtual Machine

VMM-I - Virtual Machine Management Interface

BMC - Base Board Management Controller

IPMI - Intelligent Management Platform Controller

LAN - Local Area Network

OOB-I - Out Of Band Management Interface

RMCP+ - Remote Management Control Protocol.

TOPT - Time-Based One-Time Password Algorithm

DES - Data Encryptions Standard

2 Evolution and Design of Managing Virtual Machines using Out Of Band Channel

In a typical virtualization setup, Virtual Machines and Physical servers are managed and controlled using management software. In this paper we refer to this management software as Virtual Management Software (VMS). This VMS provides advanced features such as High Availability and Live Migration. All the physical servers in data center are connected to VMS over Ethernet. In summary, VMS at an application level manages Virtual Machines (VMs) using a hypervisor that is deployed on each of the physical systems. VMS usually dedicates a

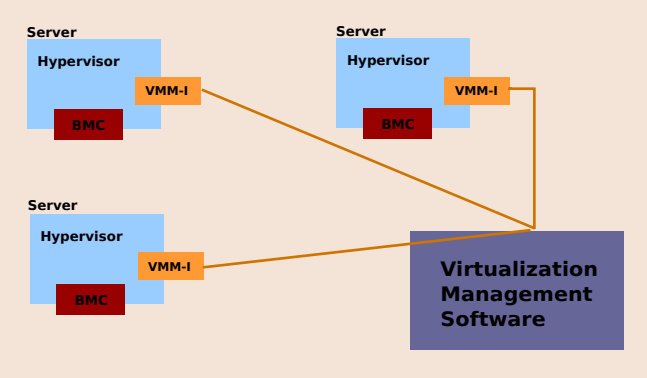

Figure 1: Current Architecture

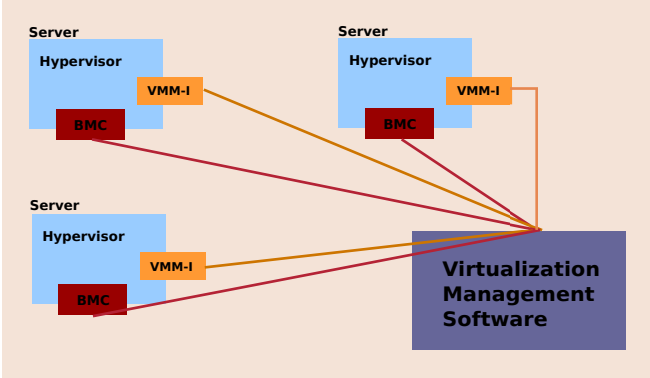

Figure 2: Proposed Architecture

system interface on each of the physical servers to communicate with the hypervisor. This interface is called as the VM Management Interface (VMM-I) and is shown in Figure 1.

The physical servers used in data centers today are mostly Enterprise grade equipped with Out Of Band Systems Management capability. One of the widely used OOB Management implementations has a Baseboard Management Controller (BMC) embedded inside the physical server. BMC supports the industry-standard Intelligent Platform Management Interface (IPMI) specification, which enables users to configure, monitor, and recover systems remotely. Each of these physical servers hosts a base hypervisor and VMs on top of it. Each of these hypervisors are connected to VMS through VMM-I.

2.1 Proposed Solution

Figure 2 depicts the proposed model, wherein we are using the Out Of Band Management Interface (OOB-I) as a secondary interface to manage the VMs using the hypervisors. Similar to VMM-I, each OOB-I is also connected to VMS. VMS uses OOB-I channel to communicate with BMC by using IPMI over LAN. IPMI OverLAN is a functionality that provides remote machines the ability to send IPMI messages over Network to BMC using UDP protocol (IPv4). The UDP packets are formatted to contain IPMI request/response messages with IPMI session headers and Remote Management control Protocol (RMCP+) headers (IPMI v2.0 Spec). For IPMI OverLAN to work, BMC needs to have a dedicated Management Network interface associated to it.

2.2 Architecture of Proposed Data Path to establish OOB communication channel between VMS and hypervisor

When VMS intends to send a message to hypervisor, it will encode the message in an IPMI message format and send this message using RPMCP+ protocol over OOB-I channel. Once sent to BMC, these messages are picked up by the hypervisor and decoded back to the original format (VMS message). Similarly when the hypervisor needs to send data to VMS, it will send it to BMC and VMS reads these messages over OOB-I.

In this solution we implemented two Buffer queues in BMC, one to hold the data that VMS sends to hypervisor and the other to temporarily store data sent from hypervisor to VMS. To access these buffers we need four new sets of IPMI commands to read and write the respective queues. Figure 3 gives an overview of the stack

Figure 3 shows the different modules that are involved in enabling the OOB-I channel between VMS and hypervisor. When VMS needs to communicate with the hypervisor over OOB-I, VMS would encode the message into an IPMI packet as payload and send it over the OOB-I. In this solution the four IPMI commands associated with the BMC Buffer Queues are implemented as OEM Commands. The idea is to carry these VMS/hypervisor messages as payload using IPMI Commands, so the encoding and decoding logic would be confined to the payload and would keep the IPMI/BMC changes at minimal. This design avoids decoding of every message VMS/hypervisor sends as a separate IPMI message.

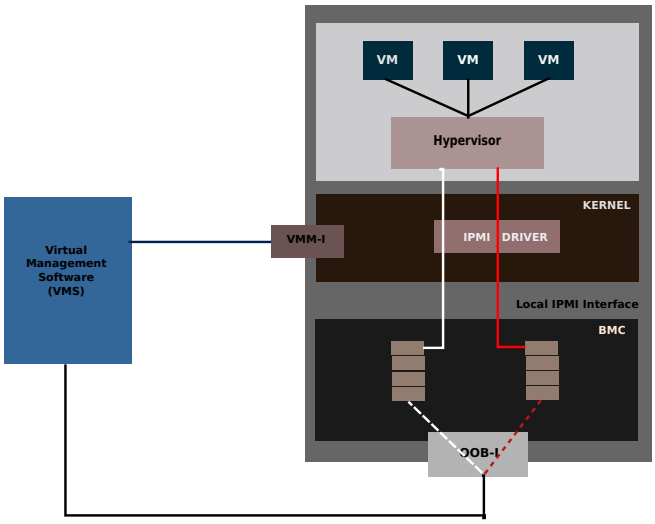

Figure 3: Proposed stacks overview

2.3 Low-level Details of the Implementation

Consider a scenario when VMS decides to communicate with a hypervisor. VMS would encode the message that needs to be sent to the hypervisor in IPMI format and use the new IPMI commands to send this message to the appropriate BMC. For VMS to communicate with hypervisor it needs to know the IP address of OOB-I apart from VMM-I IP. VMS message structure that needs to be sent to hypervisor.

```
struct vms_message {
cmd_id  cmd,   //actual command id
vm_id vm_name, //name of virtual machine
ip_addr oob_i, //IP address of OOB-I
ip_addr vmm_i, //IP address of VMM-I
.              //meta data
.
};
```

API to convert vms_message into raw IPMI format

```
char *ipmi_payload  convert_to_raw_ipmi\
_data(struct vms_message *vms_data) {
// returns *ipmi_payload - VMS message
// converted into IPMI format
};
```

API to send the vms_message to BMC over OOB-I using RMCP+ protocol. IPMI over LAN requires user authentication.

```
char ipmi_send_message (char *ipmi_\
payload, structipmi_system_interfac\
e_addr bmc_addr )\
{
  .
  .
 char *user_name,
 char *passwd,

  Request message
  [NetFn]
  [CMD ]   -  [COPY_VMS_TO_BMC_BUFF_1]
  [Payload ] - [ipmi_payload]
  char *ipmi_payload
  .
  .
};
```

BMC on receiving the IPMI message from VMS, saves the payload in Buffer Queue1 (Figure 3). hypervisor periodically check for any VMS requests in Buffer Queue1. Any messages in this Queue are picked up by hypervisor using the new IPMI command. The next task is to decode the payload message from BMC, which is exactly the reverse of the encoding mechanism that was carried out in VMS.

IPMI command to read message from Buffer Queue1

Request

```
[NetFn] [Cmd] [OEM_ID] [READ_BMC_BUFF\
_QUEUE1]
```

Response

```
[Completion Code ] [ Payload ]
```

API to read the vms_message to BMC (Queue1) over local IPMI interface.

```
char *ipmi_payload  ipmi_get_message (\
struct ipmi_system_interface_addr bmc_\
addr )   {
  .
  .
  Read Payload from BMC-Queue1
  [NetFn]
  [CMD ]   -  [READ_BMC_BUFF_ QUEUE1]
   .
   .
   Response message
   [Completion Code ]
   [ IPMI_Payload ]
```

```
   return ipmi_payload
};
```

API to decode the ipmi_payload into original VMS message format

```
struct vms_message *vms_data  \
convert_to_vms_format (ch\
ar *ipmi_payload  )
```

3 Out of Band Systems Management with Virtualization - Challenges

Out of band management of VMs brings additional challenges to the table, some of which are discussed below. In section 3.1 we discuss how to secure the OOB-I channel end to end, i.e. starting from VMS to hypervisor. Then in section 3.2 we discuss how to ensure that only a rightful authority can initiate state changes to the virtual machines. Both scenarios will be explored using a Linux/hypervisor case study.

3.1 Securing the OOB-I communication Channel

The complete communication channel for the proposed solution comprises of multitude of components, as illustrated in Figure 4.

- OOB-I
- BMC layer
- Linux Kernel Layer
- Exposed Userspace IPMI device
- Userspace hypervisor software

In the above mentioned components, OOB-I is generally secured using RMCP+ protocol. BMC access is typically controlled by a password authentication. Linux kernel space is off-limits to user space processes and hence is considered secure. In this implementation the modified hypervisor software uses the IPMI Device interface (`/dev/ipmi`) to obtain the information from BMC. So any stray read/write to BMC through this device can reveal the payload.

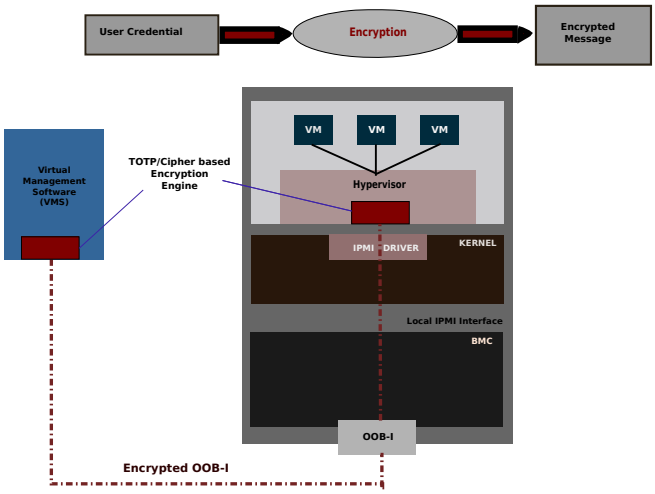

Figure 4: TOTP-Cipher Encryption

We used "time-based one-time password Algorithm" (TOPT, RFC6238) signature engine to generate a secret key within VMS. This secret key is used as auxiliary input to a symmetric key algorithm based cipher (DES/AES) to encrypt the outgoing payload. This encrypted payload is then transferred through OOBI. A stray read/write to BMC cannot decode the payload since it does not have the TOPT signature.

The hypervisor layer also has the TOPT signature engine with same algorithm, which it uses internally to decrypt and verify the payload it received from BMC. Once the hypervisor has decrypted the payload, it separates the user credential, associated VM-ID and control message.

3.2 Protection against accidental state change

In a traditional virtualization setup, the hypervisor has all the authority to carry out management tasks on all virtual machines. This allows the hypervisor to shut down, close or change the state of VMs without any restriction. However, with the growing level of (mission-critical) work load running on VMs, it is important that there should be additional security layer to help avoid any accidental state change in VMs. We investigated this problem and proposed a *request-challenge-verify* interface between hypervisor and VMs.

In the solution, whenever a state change request is given to VM, the request is verified against an authorization list. Here it checks that if state change request coming from an authorized process or not. If the request is valid,

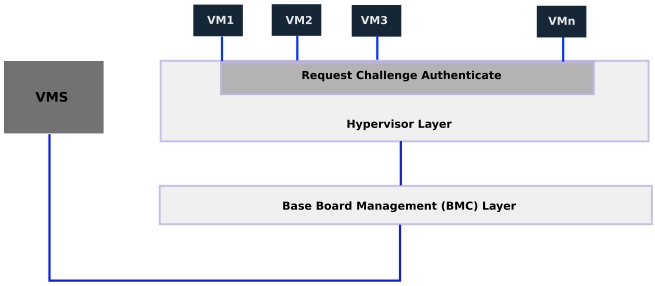

Figure 5: Request-challenge-authenticate

then it is accepted and state change activity is carried out, otherwise the hypervisor rejects the state change request and sends an alert to a registered user to inform about unauthorized attempt to change the state of a VM. This is illustrated in Figure 5.

4 Known Constraints

This implementation needs changes in VMS, hypervisor and BMC. In the absence of VMM-I, the scope of network critical tasks such as Live migration over OOB-I is limited.

5 Conclusion

We explored a method in which Systems management capability can be used to handle virtual machines on the servers and perform various tasks in the same way it would be done on a physical server. We also explored use cases in terms of handling security situations which are evolving in a virtualized data center environment. The finding here is that in a virtualized environment were each VM can be running different tasks, it is unsafe to have unquestioned authority resting with VMS, since any error can prove very costly. The other aspect is the protection of data that travels from VMS to hypervisor. These challenges are unique and they need special attention in the virtualized data centers. One important finding is that, we need further explore what are the repercussions of creating a request-challenge-authenticate framework since VMS does not enjoy super user status any more.

6 References / Additional Resources

IPMI Home Page http://www.intel.com/design/servers/ipmi/index.htm

IPMI SPEC http://download.intel.com/design/servers/ipmi/IPMI2_0E4_Markup_061209.pdf

LibVirt Virtualization http://libvirt.org

Kernel Based Virtual Machine http://www.linux-kvm.org/page/Main_Page

TOTP: Time-Based One-Time Password Algorithm http://tools.ietf.org/html/rfc6238