Secure Migration of Virtual Machine (SV2M) in Cloud Federation using Enhanced Key Management

By
Naveed Ahmad
2012-NUST-MS-CCS-031

Supervisor
Dr. Muhammad Awais Shibli
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Computer and Communication Security (MS CCS)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(November, 2014)

بسم الله الرحمن الرحيم

Approval

It is certified that the contents and form of the thesis entitled "**Secure Migration of Virtual Machine (SV2M) in Cloud Federation using Enhanced Key Management**" submitted by **Naveed Ahmad** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Muhammad Awais Shibli**
Signature: ____________________

Date: ____________________

Committee Member 1: **Dr. Abdul Ghafoor**

Signature: ____________________
Date: ____________________

Committee Member 2: **Dr. Zahid Anwar**

Signature: ____________________
Date: ____________________

Committee Member 3: **Ms. Hirra Anwar**

Signature: ____________________
Date: ____________________

I dedicate my dissertation work to my loving family, especially my mother,
Badr-un-Nisa
for her love, never-ending support and encouragement in masters degree.

Certificate of Originality

I hereby declare that this submission titled **Secure Migration of Virtual Machine (SV2M) in Cloud Federation using Enhanced Key Management** is my own work.To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged. I also verified the originality of contents through plagiarism software.

Author Name: **Naveed Ahmad**
Signature: ________________

Acknowledgment

First and foremost, I would like to thank **Allah (SWT)** for blessing me with the strength, courage and wisdom to complete my MS thesis. I would like to express my deep gratitude to my parents for their prayers, everlasting love, encouragement and support that led me towards the achievement of this worthwhile goal. More than anyone, I am greatly indebted to my mother for her support, patience and encouragement.

I am highly honored to be mentored and supervised by *Dr. Muhammad Awais Shibli.* Dr. Shibli has always been a great source of motivation and inspiration. I am grateful for his advice that helped me get to where I am today and look forward to his guidance in the future as well.

I offer a special thanks to my committee members, who have guided me in many different ways. I am grateful to *Dr. Abdul Ghafoor*, for his valuable time and comments on my research work. Thanks to *Dr. Zahid Anwar* for useful guidelines regarding the completion of MS thesis work. Many thanks to *Ms. Hirra Anwar* for numerous thought-provoking conversations and suggestions during the implementation phase.

Finally, I would like to thank *Ms. Ayesha Kanwal* & *Ms. Rahat Masood*, for their support and guidance during research work. I really appreciate your help and encouragement that you have always extended towards me whenever I lost hope and self-confidence.I would like to thank *Ms. Yumna Ghazi* & *Azeem Saeed*, for their valuable reviews and suggestion for conference & journal papers.Lastly, I would like to thank my senior *Muhammad Kazim* and my class fellow *Fowz Masood* for healthy research discussions and suggestions, we all had a really great and memorable time at KTH-AIS lab.

Naveed Ahmad

Table of Contents

List of Figures

List of Tables

Abstract

Virtual Machine (VM) migration is one of the core features of Cloud, which is mainly used for high availability, workload balancing, hardware maintenance and fault takeover. It is used by private Cloud Service Providers (CSPs) for the migration of VM on the public Cloud when the demand for computational resources increases. However, VM migration has severe security issues and it is prone to active and passive attacks. By exploiting the vulnerabilities of the migration process, attacker can launch attacks on availability, integrity and confidentiality of the VM data by illegally accessing or adding malicious code to VM images. Therefore, CSPs are reluctant to use this important feature, especially when the VM contains sensitive data. Previously, security aspects in the VM migration process were not completely explored; therefore, this paper proposes a comprehensive system for secure migration of VM in the Cloud environment which provides security features such as authorization, confidentiality, replay protection, integrity, mutual authentication and source non-repudiation with negligible modifications in existing infrastructure. We have carried out a thorough security requirement analysis of the VM migration process. We have enhanced the key manager of Cloud provider which now offers new features for the management and storage of keys involved in the SV2M solution. In addition, we have also incorporated the SV2M in a widely-used open source Cloud platform, and have evaluated the SV2M system with respect to performance and security.

Chapter 1

Introduction

Chapter 1 is intended to provide a comprehensive overview and literature review on Virtualization, Cloud Computing and security challenges of Virtualization features in Cloud. It also provide the motivation of our thesis work which is based on the one of the feature of Virtualization technology ,VM migration and its use in traditional as well as in Cloud environments. The motivation part is categorized into three main sections: Virtualization and VM, VM in IaaS delivery model of Cloud , and VM migration between Cloud Services Providers (CSPs). It paves the way for rest of the thesis and defines the overall thesis structure. We briefly state our aims for the thesis and define the scope within the boundary of Secure VM migration. The chapter concludes with the details regarding the structuring of thesis work and by highlighting the goals of each chapter.

1.1 Virtualization: Background

Virtualization technology is the abstraction of hardware resources to enable improved sharing between multiple clients. There are different types of virtualization such as hardware virtualization, operating system virtualization and application virtualization. Virtual Machine Monitors (VMMs) and the Virtual Machines (VMs) are two important components of virtualized environment [1]. A VM is defined as **an efficient isolated clone of a real hardware machine**. VMM is defined as a layer of software/tool which satisfy the following three properties:

1. A VMM must provides VMs with same essentially same functionalities to the real world hardware machine.

2. Virtualized environment may degrades the performance of programs when they execute in a VM rather than on the real machine.

3. The VMM must completely controls and isolate the system resources.

VMMs are classified into two categories based on their usage. In type I VMM's, which are also known as hypervisors, directly run on the bare hardware while type II VMMs run on the host operating system(OS) [2]. There are several commercial and open-source hypervisors are available including VMware's vSphere Hypervisor (ESXi)and the Xen hypervisor. The basic concept of virtualization is shown in Figure 1.1

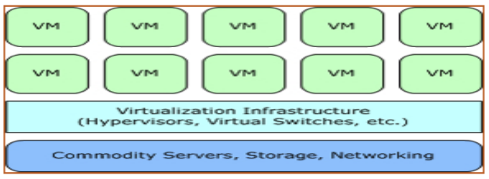

VM
Virtualization Infrastructure (Hypervisors, Virtual Switches, etc.)
Commodity Servers, Storage, Networking

Figure 1.1: Virtualization Technology.

1.2 Benefits of Virtual Machine (VM)

Before the Cloud technology, VM was mostly used by administrator in the traditional data centers. There are many benefits of VM in IT infrastructure.Few of them are mention below.

1. Virtual Machine helps in making efficient/better use of hardware resources in data centers.

2. It facilitates a greater degree of abstraction.

3. It is easily moved from one piece of hardware to another. VM migration features is commonly used for load balancing in data centers

4. It is used to create more scale able and flexible infrastructure.

1.3 Security Challenges of Virtualization Technology

Besides the several benefits of virtualization, there are several security challenges associated with it. This section discusses different security issue of virtualization technology in Cloud environment.

1.3.1 Security Issues of Hypervisor

An attacker can install malicious OS in the allocated VM which may compromises the hypervisor by modifying the source code to gain access to sensitive contents of memory [3]. In hyperjacking technique attacker install malicious program to take control of underlying OS [4].

In VM Escape attack, a program running in one VM can get root access to the host machine. Escaping the guest OS allows the malicious VM to interact with the hypervisor and provides them access to other guest OS on the system as well. Complete control of the underlying operating system by hiding itself from administrator and security software [5].

1.3.2 Security Issues of VM

If isolation is not properly implemented in virtualized environment, attacker can use covert channels for unauthorized communication with other VMs of the system. Guest OSes of VM can be target with the same attacks on OSes of physical systems to compromise them. Trojans and malwares can be used by attacker for traffic monitoring, stealing critical data, and tampering the functionality of VMs.Viruses and worms which exploit the mistakes of the buggy software used to take control of VMs [6].

Furthermore, zero day attacks can be launched on un patched OS of VM. Attacks on TCP/IP layers such as sync attack can be launched on a Cloud network with little effort. Furthermore, the saved/paused state of guest virtual machine (suspended or paused VM) as a disk file store in plaintext . Attacker can compromise the integrity and confidentiality of the saved VM state and when restored VM may not function as desired. VMs are also migrated in traditional data centers for load balancing and fault takeover. The state of VM memory stored in file and attacker can modify the content of VM during migration which is severe security threat. VM migration is also in Cloud environment [7].

1.4 Cloud Computing

Cloud computing is gaining attention in small and medium enterprises (SME's). It has revolutionized the IT industry in the last decade and it is steadily gaining attention in the business world because of scalability, increased efficiency, lower infrastructural cost and better utilization of hardware resources. It is combination of several other technologies such as Virtualization, Service Oriented Architecture (SOA) and web 2.0 [8]. Cloud Computing is providing many benefits to SME's, however there are still many significant barriers in its adoption. The security of data and information in Cloud is the main concern of any organization.

1.5 Characteristics of Cloud

According to *National Institute of Standards and Technology (NIST)* definition of Cloud, it's consist of five necessary characteristics (as shown in Figure 1.2) which make it different from traditional computing environments [9]. These characteristic of Cloud have positively improved the Cloud usage and integration within the IT industry.

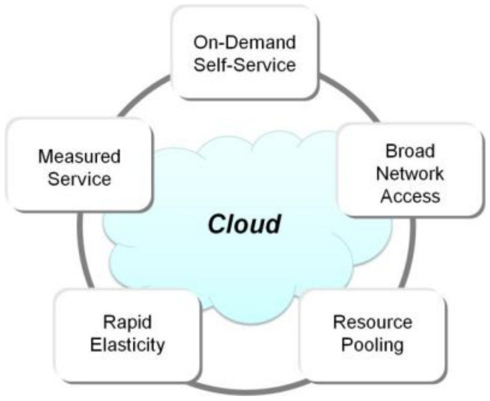


Figure 1.2: Characteristics of Cloud.

1. On-demand Self-Service: Cloud users can avail CSP services as per their demands. They can change the computing capabilities automatically without requiring any intervention of CSP.

2. Broad Network Access: Extensive network services provided by CSP are used by mobile phones and laptops through standard mechanisms.

3. Resource Pooling - Virtualized computing resources (such as storage,network bandwidth, and virtual machines) are provided by the CSP to serve multiple Cloud users in multi-tenant model.

4. Rapid elasticity - Cloud provides the quick and elastic provisioning of virtualized resources.

5. Measured service - In Cloud services model, each and every thing is monitored, controlled and reported such as resource usage for transparency between CSP and consumers.

Cloud Service & Deployment Models: Cloud computing is also classified based on their data and services geographic location. Cloud, computing architecture is service oriented which is capable of offering any thing *"as-a-service"* such as Software-as-a-service (SaaS), Platform-as-a-service (PaaS), Infrastructure-as-a-service (IaaS).There are many other such Security-as-a-service, Database-as-a-services etc. There are many companies which are offering on-demand and low-cost services to the the consumers such as Amazon(PaaS and IaaS),Mircosoft(PaaS), RackSpace (IaaS) and Salesforce, Google (SaaS). There are four deployment models of Cloud including public, private, community and hybrid Cloud. Each deployment model has its own pros and cons in terms of security and functionality they can offer. However our interest is on IaaS service model of Cloud. From the Cloud service provider (CSP) perspective, both delivery models SaaS and PaaS are dependent on IaaS for their services. Similarly any security violation in IaaS delivery model will have an effect on SaaS and PaaS security and vice versa [8].

1.6 Motivation

The motivation section is categorized into three different but interrelated domains, which include *Virtualization and virtual machine*, *VM in IaaS delivery model of Cloud* and *Migration of VM between CSPs.* In the following paragraphs, we present the details regarding each one of them.

Virtualization and Virtual Machine: Virtualization is the primary component that makes Cloud computing special. Virtualization enables a single system to run multiple isolated virtual machines (VMs), operating systems or multiple instances of a single operating system (OS). The concept Cloud is merger of several other technologies such as virtualization and Service Oriented Architecture (SOA). Virtualization is one of the critical and important technologies of Cloud and it enables the abstraction of hardware resources for the purpose of improved and better hardware utilization leads to reduce operational and investment costs. In the past, virtualization technology used in the traditional data center for better utilization of organization resources. However in Cloud environment, CSPs provision virtualized computing capacity (VM) as per customer needs [8, 10].

VM in IaaS delivery model of Cloud: Virtual Machine is important components of IT infrastructure. CSPs allocates VM to consumers from the pool of virtualized computing resources such as storage, network, processing and others in IaaS delivery model. Security constraints of virtualization technology are also inherited in the Cloud environment along with their benefits. Security of VMs becomes critical for the overall security of Cloud because virtualization technology introduces new attacks and challenges [10, 11].

Migration of VM between CSPs: VM migration or mobility is the key feature of virtualization technology which is used to provide hardware/system maintenance, work load balancing, and transparent management in conventional data centers as well as in Cloud infrastructure [12]. This useful feature is used by the Private Cloud for the migration of VM on the Public Cloud when the demand increases for computing. Apart from providing major features, migration provided by KVM, VMware, XEN and Hyper-V hypervisors is prone to severe security risks. VM migration is not secure because of unavailability of strong security features in hypervisors. VM migration without security becomes single point of failure for Cloud environment because intruder can inject malicious code or modify the VM content [13].

1.7 Aims and Scope

In this thesis we aim to identify and address the important security issues of Virtualization technology which effects the infrastructure as a service (IaaS) delivery model of Cloud. To achieve this goal, first we will extensively explore the vast domain of Virtualizaiton and its features which are used in Cloud environment, highlight its security issues and its effects on cloud technology. Finally aim to design and implement a secure secure VM migration (SV2M) system that meets the pressing needs of CSPs. Our scope, therefore, is limited to following research objectives:

Objective 1: To provide an assessment criterion for the evaluation of secure VM migration systems . This objective can be achieved by carrying out an extensive literature survey in the domain of Virtualization and Cloud from the perspective of both security and functionality.We intend to use this work as a tool to evaluate many state-of-the-art secure solutions for VM migration in traditional data enters as well as in Cloud environment.

Objective 2: To design and develop holistic solution for the secure migration of VM between Cloud which fulfills the our assessment criteria , provides the strong security features and integrated with the open source Cloud platform OpenStack. .

1.8 Research Contributions

The primary contributions related to our research field are given below:

- **Survey on Secure Live Virtual Machine (VM) Migration in Cloud:** After extensive literature review, we have identified threats and vulnerabilities in virtual machine migration process. An assessment criterion has been devised that every secure VM migration system must fulfill it. In addition to this, We have made a contribution to this part by identifying certain features performing an extensive literature survey on the security issues of secure VM migration solutions. We have further performed a comprehensive analysis of existing secure VM migration approaches given in literature and evaluated them according to our assessment criterion [14].

- **Secure VM Migration (SV2M)in Cloud Federation:** A secure VM migration system has been designed and developed which provides security features such as authorization, confidentiality, replay protection, integrity, mutual authentication and non-repudiation with negligible modifications in existing infrastructure We have used secure tropos methodology for the representation of threats, vulnerabilities on migration process and possible security control against them. We have enhanced the key manager of Cloud Provider which offers new features for management and storage of keys involved in SV2M solution. In addition to this, we have incorporated the SV2M in open source Cloud platform Open Stack, which is used by large research communities and CSPs. We have done security evaluation of SV2M system using AVISPA tool. We also evaluated the performance of our solution using time utility which is available in the linux OS [15].

1.9 Limitations

In order to limit the scope of this work, we have restricted our research attention to the security of migration process in Cloud domains. Therefore, our research contribution focuses on providing security features (confidentiality, integrity, mutual authentication,authorization, reply protection and source non-repudiation) in inter-Cloud VM migration ; other security concerns such as trust establishment between Cloud etc. remains the foremost problems. Since, these security issues are not directly related to our work so are understandably considered beyond the scope of this thesis work.

1.10 Organization of Thesis

Our thesis is organized in the form of well-organized chapters, where each chapter is directed to provide valuable insights that positively lead towards the completion of this thesis work. A brief overview of our thesis road-map is shown in the Figure 1.3.

- *Chapter 01* provides a comprehensive overview of virtualization and Cloud technology, their features and issues of security in Cloud due to virtualization . It also provides the motivation of our thesis work which is based on the useful feature virtual machine (VM) migration of virtualization technology and its security requirements. It explains in detail the main areas of research which are focused. We briefly state our aims and give details regarding the contributions which have been made throughout this thesis.

- *Chapter 2* discusses the details regarding the research approach that we have assumed to achieve our thesis objectives. We have explored various research methods and decide to follow mixed research methodology. Research approaches including deductive, qualitative, empirical and analytical, have been used to achieve the our research objectives.

- *Chapter 3* explains our proposed assessment criteria which is based on potential security features that are positively imminent for the evaluation of secure VM migration in Cloud. Further, it gives details regarding the proposed criteria which has been formulated after carrying out an in-depth literature survey of security of VM migration in Cloud environment. In addition to this, comparative analysis of numerous secure VM migration systems using the proposed assessment criteria is presented.

- *Chapter 4* describes the design, architecture and implementation of Secure VM migration system for Cloud environments. The enhanced architecture of Key manager which is used in our system and SV2M integration with OpenStack is also discussed. In addition to this, the complete implementation details including XML encryption/decryption, XML signature, Python/bash code snippets are also presented in this chapter.

- *Chapter 5* describes the details of our thesis evaluation with respect to security and performance. Validation of security features has been carried out using AVISPA, which is a security protocol verification

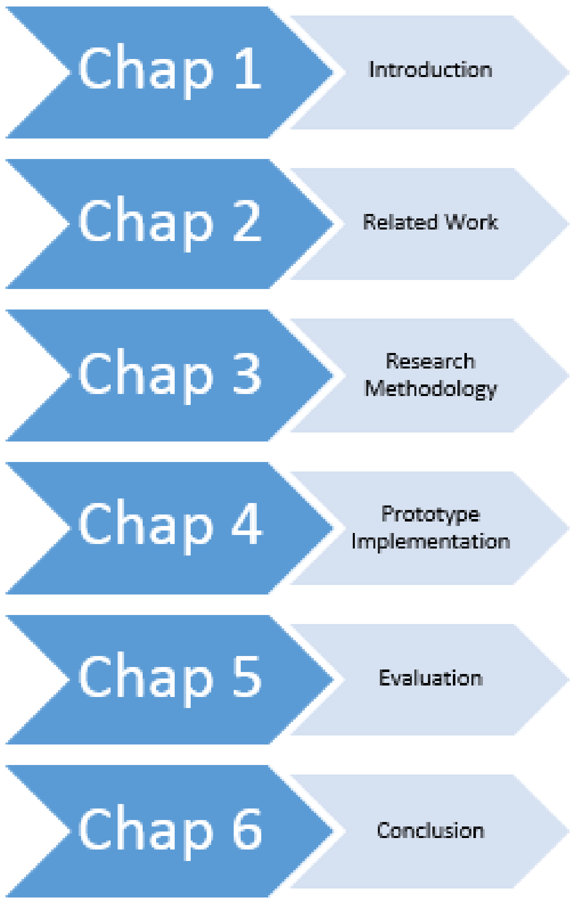


Figure 1.3: Organization of Thesis.

tool.Further, performance test of insecure and secure VM migration also carried out in OpenStack Cloud using time utility of linux which is used to monitor the system resources.

- *Chapter 6* presents the summary of our research findings and concludes by stating the future research directions of our thesis. It highlights the vital research problems of virtual resources security in Cloud which are still needed to be resolved particularly the issues related to security.

Chapter 2

Related Work

This chapter explains the related work done in the area of security of VM in traditional data centers and Cloud environment. Cloud computing has been a important area of research since 2000 and a lot of research has been done in the area of use of virtualization in Cloud. In this research work , we have done literature review of related work to secure vm migration and survey of existing solutions/approaches. We have followed the layered approach to review the literature of securing the virtual machines in Cloud. The first part of the chapter presents about the VM migration and its different types which are supported by Open sources and commercials hypervisors.The second part presents the identified threats and vulnerabilities in VM migration process, identified security requirement for secure VM migration solutions and comparison of existing solution/approaches with respect to security features. The last part of this chapter presents the secure tropos methodology for security requirements of VM migration and also security reference diagram for the representation of threats/vulnerabilities.

2.1 About VM Migration

This section explains about the VM migration, its different types and support in KVM, XEN and VMware Hypervisors. VM Migration feature is one of useful features of virtualization technology which is used for migrating a VM from one physical server to another or from one data centre to another. This feature provides efficient system maintenance, load balancing and proactive fault tolerance in enterprises infrastructures [12, 13]. VM migration is also used in Cloud Federation to provide cloud bursting feature [16].

2.1.1 VM Migration Types:

VM migration is classified into following types due to its usage in traditional data centers.

Cold Migration: In Cold migration, first VM is shut down and then migrated to other destination host, it is also known as offline migration [17].

Hot Migration: In hot migration, VM is migrated without shutting down the machine and it is also used to minimize the downtime of the services running inside the VM. Live and suspend/paused VM migrations are types of hot migration.Live migration is defined as transfers of running VMs from one physical server to another with minimum downtime and without interrupting the services running inside VM [12]. Live migration is further classified into memory migration and block/storage migration.

Live Migration: In memory migration only contents of volatile memory of VM are migrated and in block migration, the storage of VM is also migrated along with memory and it takes longer time as compared to memory migration.

Suspend/Pause Migration: However in Suspend/Pause migration technique, contents of VM stores in disk or in memory (RAM) respectively before transfer from one cloud to another [18].

2.1.2 VM migration support in Hypervisors:

VM migration is useful and important feature so that why it is supported by Xen, VMware, KVM and Hyper-V hypervisors. XEN hypervisor supports the suspended/live VM migration using XenMotion module and it also supports cold migration [19, 20]. KVM is another well know open source Hypervisor which is also supports both types of migration features [21]. VMware vSphere and Microsoft Hypervisor-V also support live VM migration feature [22]. Open source and commercial Cloud platforms which are using these hypervisors support VM migration in cloud environment.

2.2 Identified Threats & Vulnerabilities in VM Migration

John et al [13] has empirically demonstrated that live VM migration process is prone to active and passive attacks. Attacks on live VM process are categorized into control plane, data plane and migration module classes. This section discuses the threats and vulnerabilities on migration process.

Control Plane: Hypervisor operations such as initiation and management of live VM migration must be authenticated and resistant against tampering. Furthermore, protection against spoofing and replays attack should be provided. We have identified various vulnerabilities and threats on control plane which are mentioned in Table 2.1 and Table 2.2 respectively.

Data Plane: Live VM Migration occurs in this plane, memory contents such as kernel states and application data transfer from one physical server to another. Attacker can use ARP spoofing or DNS poisoning techniques to launch man in the middle (MITM) attack on insecure communication channel. This introduces active and passive attacks during the migration process. Therefore secure and protected channel must be use to minimize snooping and tampering attempts on migration data. We have identified various vulnerabilities and threats on data plane which are mentioned in Table 2.1 and Table 2.2 respectively.

Migration Module: VM Migration functionality of VMM is implemented by software component which is known as the migration module. Vulnerabilities in migration module may allow attacker to compromise the VMM and any guest OSes as well. Possible threats due to vulnerabilities in migration module are shown in Table 2.2. Vulnerabilities in migration module are shown in Table 2.1.

Table 2.1: Identified Vulnerabilities in Live VM Migration

Vulnerabilities	Description
Vulnerabilities in Migration Module	Loop holes/bugs in migration module of hypervisor such as following vulnerabilities in migration restoration routine of Xen. 1. Stack overflow due to integer signedness issue. 2. Heap overflow due to issue in memory allocation routine.. Aforementioned bugs may allow attacker to achieve privileged code execution and completely compromise the VMM and host machine [3].
Vulnerabilities in live VM Migration process (control and data plane)	Insecure live VM migration provided by Xen, KVM and VMware. It is vulnerable to severe security attacks

Table 2.2: Identified Threats in Live VM Migration

Threats	Description
Complete Control of VMM or Guest OS (Migration Module)	Attacker may gain full control of VMM or host due to stack, heap and integer overflow vulnerabilities in migration module.
Lack of access control polices in VM migration (control plane)	It may allow an unauthorized user to initiate and migrate the VM. Attacker can perform following action. 1. Denial of service by initiating a large number of outgoing migrations of VM's to a legitimate victim VMM. 2. Migration of VM with malicious code for example malware/ Trojans to legitimate VMM. To gain control, malicious VM can be used to launch attack on VMM and other guest VM. 3. Attacker first initiates unauthorized incoming migration request to influence others to transfer VM's. When VM is migrated, attacker gains control of guest VM by migrating it on his machine. 4. Attacker can falsely advertise resources of the compromised platform to influence others to share workload by migrating VMs on compromise machine.

Continued on next page

Table 2.2 – *Continued from previous page*

Insecure communication channel (data plane)	Attacker can launch active and passive attacks during VM migration on insecure and unprotected channel. 1. *Passive Attacks:* 1.1. Leakage of sensitive data from memory of migration VM such as cryptographic keys, password, application data. 2.2. Capturing authenticated packets and replay latter. 3.3. Attacker can identify the guest VM's based on size and duration characteristics of migration. 4.4. Attacker can also identify the source and destination of migration process. 2. *Active Attacks:* 1.1. Manipulating memory of VM during live migration 2.2. Manipulating specifics applications in memory of migrated VM for example authentication services such as SSHd, Pluggable authentication module (PAM) in live VM migration
Migration of low security level VM near high security levels VM	Placement of low security level VM near high security level VM during migration process. Attacker can launch attack on high security level VM by exploiting low security level VM [12].

2.3 Identified Security Requirements

This section presents the our identified and formulated essential security requirements which should be supported by secure live and offline VM migration solution in Cloud environment. The detail of each requirement is given below.

1. ***Integrity Verification of Platform:***

   The destination platform cryptographically identifies itself to source for trust establishment.

2. ***Mutual Authentication:***

Attacker can launch MITM attack using technique such as route hijacking or ARP poisoning in the migration process. In order to avoid MITM attacks on live VM migration, source and destination platforms must mutually authenticate each other.

3. ***Authorization (Access control policies):***

   Appropriate access control policies must be provided to secure the live VM migration process. An unauthorized user/role may launch VM initiate, migration operation. Unauthorized activities can be prevented using access control list (ACL's) in hypervisor.

4. ***Confidentiality and Integrity of VM during migration:***

   An encrypted channel must be established so that an attacker can not get any information from VM contents and modification of contents can be properly detected. This will help to avoid active attacks such as memory manipulation on live migration and passive attacks such as leakage of sensitive information.

5. ***Replay Resistance:***

   Attacker can capture traffic and replay it latter to get authenticated in VM migration process. Therefore live VM migration process should be replay resistant. Nonce's can be used to prevent replay attack in migration.

6. ***Source Non-Repudiation:***

   Source host cannot deny from VM migration operation This feature can be achieved by using Public key certificate.

2.4 Analysis of Existing Approaches & Solutions

This section discusses in more detail some of the existing solutions or approaches for secure VM migration in cloud environment. Many of existing solution are using Trusted Platform Module (TPM) in their solution and only support offline migration. TPM dependent solutions require changes in software (virtualization of TPM in hypervisor) and hardware of current Cloud infrastructure. Approaches which provide security in live VM migration are not comprehensive and do not fulfil all the essential security features (such as Mutual Authentication, Authorization, Replay Resistance , Confidentiality and Integrity of VM contents during migration process and Non-Repudiation) of Live Migration process [12, 23].

Isolated/segregated migration uses Virtual LAN (VLAN) to isolate migration traffic from other network traffic because it reduces the risk of exposure. However it does not provide any security feature and cost of VLAN management is also linked with population of VM's [12, 24]. In Network Security Engine-Hypervisor based approach, firewall and IDS/IPS functionalities become part of hypervisor for protection against intrusions. However, it does not provide security features during the VM migration process [12].

In Policy or Role based secure migration approach, only offline VM migration is supported and it requires changes at software and hardware level for linking in existing infrastructure [12, 25]. Security requirements such as mutual authentication, replay resistance and source non-repudiation are not part of this approach.

Secure VM-vTPM migration protocol provides mutual authentication and establishes secure session for subsequent communication. Remote verification is performed by the source to check or verify the integrity of destination system. However, this approach does not support live migration and keys of vTPM are also stored outside the TPM, therefore prone to leakage and unauthorized modification [12, 26].

Another such approach includes vTPM migration protocol, where both parties mutually authenticate each other and then property based remote attestation is performed by source host to verify the integrity of destination. Key exchange is performed using Diffie-Hellman (DH), which is used to achieve confidentiality during migration process. This approach only supports offline migration and vTPM state is also migrated [23].

In this paper, author proposed solution that uses proxy and secure shell (SSH) to ensure security while VM migration. Proxies are used to restrict access to end nodes which are involved in VM mobility and SSH tunnel is established between both proxies to achieve confidentiality. However it does not provide authorization and non-repudiation security features and port forwarding is require on all intermediate devices and firewalls [16].

In this paper, author proposed new architecture for cloud platform, which is known as Trusted Cloud Security Level (TCSL). It uses set of policies to customize Cloud zones and reliable migration module for VM migration in trusted Cloud. However it does not provide any security feature during the migration process. In TCSL approach, VM isolation is based on its security level in trusted zone of Cloud. It requires changes in existing Cloud platform for availability of security features [27].

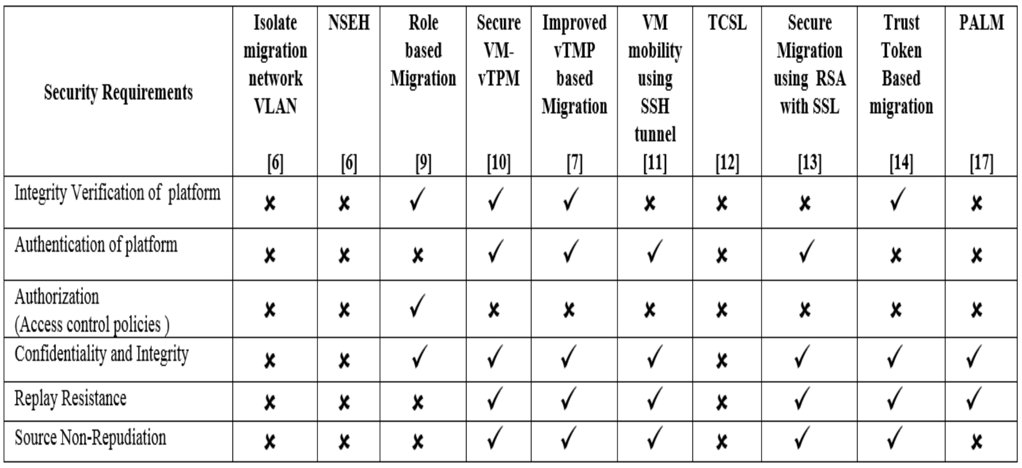
RSA with secure shell (SSL) based approach provides authentication, encryption, and reply resistance during migration process. For authentication, it requires public keys of all other hypervisors for VM migration however it includes the difficulties of public key management [28].

Trusted Token (TT) based migration technique uses satisfactory Trust Assurance Level (TAL) value of the target cloud platform for VM transfer. Platform Trust Assurance Authority considers hardware and software components of platform while issuing TT credential. TAL value is enclosed in TT credential, which is further used in VM migration. User can perform VM migration if TAL value of recipient platform is acceptable corresponding to migration policy of sender. This approach also uses the TPM in key pairing and does not provide live VM migration. Moreover, it gets complex when the multiple number of users simultaneously perform VM migration in Cloud. TAL value is dependent on hardware and software components of Cloud infrastructure; therefore change in any component (addition of new hardware, installation of new software) requires new TAL value. The performance of this approach may degrade due to dependencies on hardware chip (TPM) which is also virtualized (vTPM) in Cloud environment so that all VM's can access it [29].

Fengzhe et al. proposed Protection Aegis for Live Migration (PALM) of VM's using customized Virtual Machine Monitor (VMM) called VMM enforced protection system which provides security to running processes in VM. In this approach, protected memory pages also transfer as metadata along VM. Furthermore it does not provide authentication and authorization security features [30].

Table 2.3 presents the result of our extensive analysis with respect to security aspects.

Table 2.3: Analysis Of Existing Solutions & Approaches



Security Requirements	Isolate migration network VLAN [6]	NSEH [6]	Role based Migration [9]	Secure VM-vTPM [10]	Improved vTMP based Migration [7]	VM mobility using SSH tunnel [11]	TCSL [12]	Secure Migration using RSA with SSL [13]	Trust Token Based migration [14]	PALM [17]
Integrity Verification of platform	✘	✘	✓	✓	✓	✘	✘	✘	✓	✘
Authentication of platform	✘	✘	✘	✓	✓	✓	✘	✓	✘	✘
Authorization (Access control policies)	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘
Confidentiality and Integrity	✘	✘	✓	✓	✓	✓	✘	✓	✓	✓
Replay Resistance	✘	✘	✘	✓	✓	✓	✘	✓	✓	✓
Source Non-Repudiation	✘	✘	✘	✓	✓	✓	✘	✓	✓	✘

2.5 Secure Tropos Methodology for Security Requirements of VM Migration

We already have performed extensive literature review and find out the threats and vulnerabilities on existing VM migration process. We also have analyzed security requirements for Secure migration process. In this section we are using secure tropos methodology to model the threats and security features of VM migration using the security reference diagram which is constructed after analysing the security requirements of system. Before the development cycle, security requirements of the system collect using security referring modelling. It involves the identification of security needs/feature of the system, problems related to the security of the system, such as threats and vulnerabilities as well as possible solutions to the these security problems. Tropos methodology is used in the software development cycle which employs modelling concepts such as actors, soft goals, goals, resources and tasks. These concepts are also best suited to model the security requirements of the system, however the security requirements are usually expressed in natural language such as confidentiality and authentication goals [31].

Secure tropos methodology is used to consider security features at design phase of application or software. The security reference diagram of secure tropos is used for the representation of the link between security features, threats, protection objectives and security mechanisms. Security features represent that the end system must have associated features such as privacy, availability, confidentiality and integrity. Protection objectives of secure tropos represent a set of rules that contribute towards the achievement of the security features and the concept of goal is used for the modelling of protection objectives. Examples of protection objectives are cryptography and accountability. Security mechanisms or methods used for the satisfaction of the protection objectives. Security mechanisms are modelled using task concept because a security mechanism represents a precise way of satisfying a protection objective. Threats represent circumstances that have the potential to cause loss that violates the security features of the system. Examples of threats are password sniffing and eavesdropping attacks. The notations used by secure tropos for the protection objective, security mechanism, threats and security features are shown in the Figure 2.1. The notations of security reference diagram are linked with each with the help of positive and negative contribution links. A positive contribution link associates two nodes when one node helps in the fulfillment of the other and a negative contribution link indicates that a node contributes towards the denial of another node. A positive contribution link of security reference diagram is modelled as an arrow with a plus (+), which points towards the node that is fulfilled, whereas a negative contribution link is represented as an arrow with a minus (-) [31].

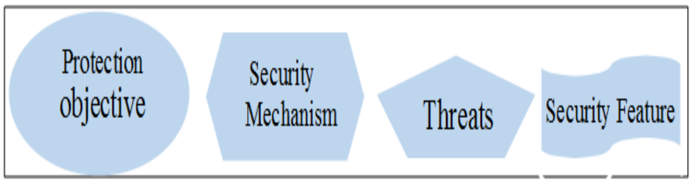


Figure 2.1: secure tropos notations for security reference diagram

VM migration in Cloud environment is vulnerable to active and passive attacks. In active attacks, attacker can modify the content or steal sensitive information of VM during migration. An unauthorized migration request may cause denial of service (DoS) attack. VM migration is prone to replay attack on unencrypted/insecure channel and attacker can use it to replay the authentication session, therefore nonces/random number are used for protection. The source of VM migration operation may deny therefore source non repudiation is required in secure migration. We have used the methodology of Secure tropos for Secure VM migration. The security reference diagram of SV2M is constructed after analyzing the security requirements which shows the threats on migration process and the possible solution against identified threats is shown in Figure 2.2. The security reference diagram consists of different symbols or notations to shows the threats, security mechanism and desirable security features of SV2M system. Threats and their possible security mechanisms are linked to security features of SV2M system using negative/positive contribution link respectively.

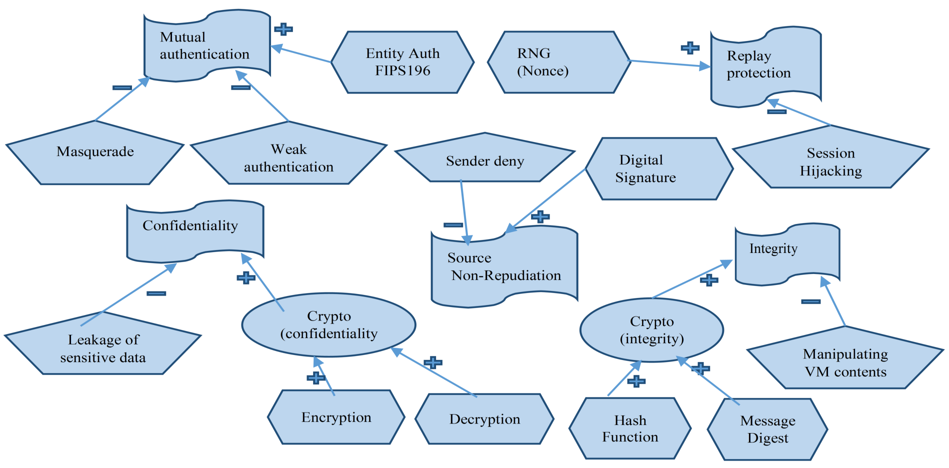


Figure 2.2: Security reference diagram using secure tropos notations.

Chapter 3

Research Methodology

This Chapter discuses the research methodology that we adopted during this thesis work to achieve our goals. This thesis work has several components; first of all the detailed study of Virtualziation technology and its usage in Cloud environment. Then security analysis of the VM migration solution to identify potential threats and vulnerabilities and also devise assessment criteria (security requirements) for different commercial and open sources solutions. After identification of security requirements , we have to design and implementation of proposed solution for the protection of VM migration process in Cloud computing. We have adopted ***deductive approach*** *&* ***research****methodologies for the completion of our research work. This chapter presents the details about the complete methodology for defining research problem, develop hypothesis, observation and development of proposed solution based on hypothesis.*

3.1 Introduction

'Research' is define as in depth study and investigation in a specific area with an objective to search for new knowledge and get some specific information related to any particular field [32, 33]. It is carried out for the most efficient solutions of unsolved problems. Research is the keen desire for knowledge that encourages us to inquire and discover the unknowns. Every research activity generally involves a series of steps such as:

1. Identify area/field of research

2. Perform extensive literature review

3. Identify the problem

4. Develop hypothesis
5. combine and evaluate
6. Certify the result to decide whether or not they fulfill the hypothesis

3.1.1 Research Method & Research Methodology

Research Methodology is standard set of guidelines and systematic procedure which is used to conduct research in any area . It consist of following steps including:

1. Information Gathering.
2. Data Review.
3. Analysis and Development of hypothesis.
4. Deduction of Research findings [32, 34].

Research methodology covers research methods, their logic and importance of the research methods for specific scenarios. Research methods may follow a systematic or objective approach depending upon the problem area. Similarly, for the this thesis work, we have done an extensive literature review for identification of potential research questions which are either unanswered or have not given due attention. Our main objectives in this thesis include:

- Detail study of virtualization and usage of useful features in Cloud.
- Identification of security requirements for VM migration.
- An assessment criteria for secure VM migration system .
- Develop hypothesis through extensive literature review.
- Validate the hypothesis through implementation and evaluation of framework.

3.1.2 Types of Research Methods

There are different types of research methods and each one of them is suitable for specific domains and research phenomenon.

3.1.3 Applied Research

Applied research is adopted to solve real world problems which targets the business, industry or general public. In the end this research method produces detailed findings or solutions of problem [33].

3.1.4 Descriptive Research

It is a survey based research methodology and the methods which are used in this research mainly focus on survey techniques . It consist of various surveys of literature/questionnaires to find facts and It does not produce any novel contribution. The whole research is done by the evaluation of already available data to researcher. [33].

3.1.5 Analytical Research

This research method required extensive study of state-of-the-art literature with an objective to conduct comparative analysis to produce detailed results and findings. This type of research ends with report which consist of facts and finding which are consolidated after performing detailed comparative analysis of the existing solutions/approaches [33].

3.1.6 Deductive research

In this research method, researcher begins to explore the broad spectrum of information say theory about some filed (hypothesis) an that why it is also known as top-bottom approach. It consist of theory/survey, hypothesis, observations and validation. Researchers narrow down by investigating the possibilities to achieve some specific goals and objectives [35]. Any research methodology that involves theory, hypothesis, observation and confirmation phases is referred to as deductive research approach.

3.1.7 Inductive Research

It is opposite to deductive research method, It uses bottom-up strategy is used to draw generalized theories from study of experimental observations (hypothesis) [35]. Inductive research is considered unreliable in comparison to deductive research because it is not logical to assume that general theory is always correct.

3.1.8 Empirical Research

Empirical Research targets the experiments or involves observation to draw conclusions from it. It helps integrating research and practice. Researcher required to formulate a hypothesis first to perform empirical research and then validate the results using facts.

3.1.9 Qualitative Research

Qualitative research provides detail information about the research area using survey and research analysis. Information gather using this research method is considered reliable and stronger analysis conclusions. Sometimes, it helps to develop a new theory that didn't exist before [36, 37].

3.1.10 Quantitative Research

As it name indicates that it is used to reduce data to numbers and analyzed using some statistical methods. Quantitative research approach supports an exist theory using observations, surveys and analysis of records for numeric figures. Findings of quantitative research are descriptive in nature [36].

3.1.11 Conceptual Research

Conceptual research used to find the basic ideas behind a phenomenon. It is a method of research in some areas such as social sciences and psychology where the researchers explore some existing idea to develop new theory or change the existing ones [33].

3.2 Thesis Research Approach

In order to achieve our research goals we have to adopt more than one research methodology described above. We have conducted our research in three phases. In first phase we have done detail study of virtualization technology and usage of useful features in Cloud environment. We have considered VM migration feature during this phase wich is used in the data center and in the Cloud. Then We have performed the in depth literature review on security challenges of VM migration in Cloud and identified security requirements for secure VM migration solutions. This leads to us to design and develop a secure system which fulfill the our identified security requirements. we have used multiple research methodologies for the completion of different phases. Conceptual research for study the security architecture of Cloud

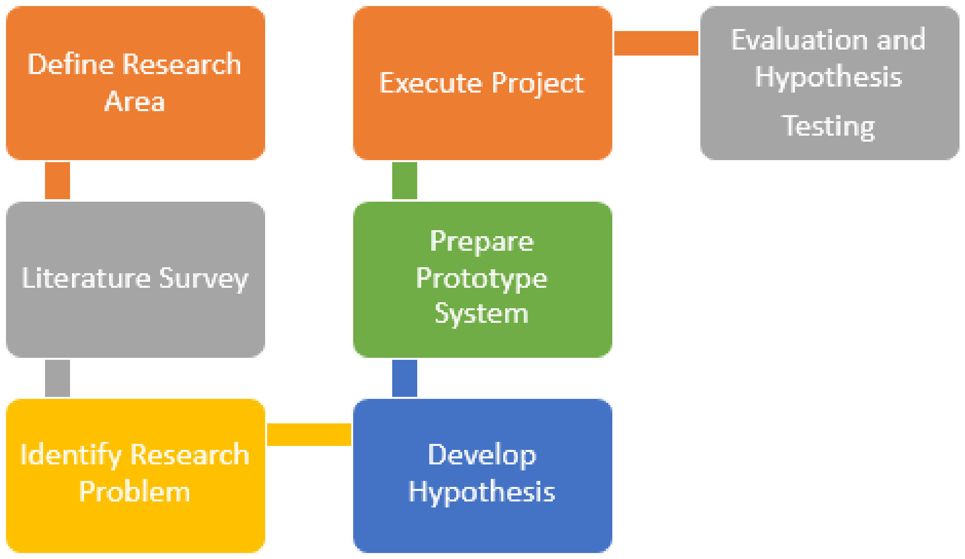


Figure 3.1: Thesis Research Workflow.

and virtualization . Deductive research methodologies for the identification of problem in Virtualization technology. Similarly, we have used empirical research to implement secure VM migration in Cloud environment.

We have taken several steps during this research for the completion of our thesis work. Figure 3.1 provides an illustrative overview of the steps of our research methodology.

3.2.1 Defining Area of Research

As a first step, VM migration in Cloud was selected as our general domain for thesis work and then we consider deductive research approach to conduct an in-depth survey on the proposed secure VM migration (SV2M)system. we have formulated our thesis research problem statement and hypothesis using facts and observations derived from this survey work. We have explored the domain of secure VM migration in Cloud and contributed into its different domains. Our first contribution involves the survey on secure VM migration in Cloud environment. An extensive research work has been conducted for the understanding of the threats and vulnerabilities on VM migration in Cloud, identification of security requirements and assessment criteria for SV2M system. In our second contribution we have proposed secure VM migration solution to resolve the identified problems of existing solutions/approaches. We have performed a detailed literature survey which covers the various security challenges and solutions from Cloud environment.

3.2.2 Literature Survey

We have chosen security of virtualization technology in Cloud as our thesis research problem statement. We have selected this as our research area after reviewing literature (research articles) and existing open source and commercial secure VM transfer solutions. Qualitative research methodology helped us in the identification of security issue in the VM migration systems.

We have used design system technique for reviewing and analyzing several security challenges associated with VM migration systems which help us for the development of our thesis research problem statement. We analyzed that there is no assessment criterion for the evaluation of SV2M system. After analyzing various solutions/approaches,we have proposed an assessment criterion and security requirements for the evaluation of secure VM migration systems.

3.2.3 Problem Statement

After conducting an extensive literature review, we have derived following problem statements regarding the security of VM migration in Cloud environment:

1. An assessment criteria is required for the evaluation of secure VM migration systems.

2. In addition to assessment criteria, there is a need to design and implement secure VM migration system which fulfills the derived security requirements and integrated with the existing infrastructure of Cloud with minimum changes.

3.2.4 Hypothesis

Analytical and deductive research methodology has helped us in establishing our thesis hypothesis. We survey various state-of-the-art Cloud identity management systems with related security issues and enlist a number of observations and assumptions which will be used to verify our hypothesis at the end. Hypothesis can be summarized in following three questions:

1. Does the insecurity in VM migration process is a major hindrance in adoption and acceptance in IT industry?

2. Is it possible to develop an assessment criteria for the evaluation of existing VM migration solutions/approaches ?

3. Is it possible to define security requirements for the secure VM migration between CSPs?

Following are the observations that we have concluded after performing an extensive literature survey:

- VM migration is useful feature which is used for load balancing and fault takeover in data center however it is single point of failure for Cloud environment.
- VM migration is vulnerable for active and passive attacks which leaks sensitive information during transfer.
- There are some open source and commercial solutions but they are not satisfying strong security features.
- Existing solutions/approaches required many changes in the infrastructure of Cloud.

3.2.5 Formulation of system design

We have specified the important modules of our system which are required to achieve objective of the thesis work. Thesis objectives include:

- Analyze features of existing secure VM migration solutions/approaches and based on these devise an assessment criteria for the evaluation of secure VM migration in Cloud.
- Design and develop secure VM migration module which easily integrated with open source Cloud platform OpenStack.

3.2.6 Execution of Thesis Research

In order to solve the problem statement and satisfy the developed hypothesis we offer In order to support the hypothesis and to provide solution to the above mentioned problems, we offer

1. Security Requirements for secure virtual machine migration systems of Cloud.
2. An implementation of Secure VM migration module for open source Cloud platform OpenStack which satisfy the identified security requirements.

To explain these arguments further, security requirements for the evaluation of VM migration solutions devised which comprises of strong security features such as mutual authentication, confidentiality etc. Finally, we have to implement secure migration module for VM using XML encryption/signature for the OpenStack.

3.2.7 Evaluation and Hypothesis Testing

We have proved the validity of our hypothesis by implementing the SV2M system for Cloud environment and then we have used AVISPA, security protocol verification tool for the evaluation phase to confirm the effectiveness of our research work. The details about the evaluation and its results are discussed in the next chapters. However, the results of our assessment shows that the our developed system provides the required strong security features, hence positively support our thesis hypothesis.

In this research work, first of all we understand and explore the domain of Virtualization and its security issues in Cloud environment. After an in-depth literature survey and analysis, we have identified the key areas of virtualization which require further attention. So Our aim is to identify and solve the potential problems of virtualization domain which also effect the Cloud.

Chapter 4

Proposed System & Implementation

This chapter presents the design, architecture and implementation of secure VM migration system for Cloud environments. Proposed system ensure the identified security requirements and required minimum changes in existing Cloud infrastructure.SV2M system introduces strong security features such as mutual authentication between CSPs, authorization before initiation of migration operation, confidentiality, integrity, replay resistance and non-repudiation.We have used open source libraries (Mcrypto,Pyssl) and Cloud platform software (OpenStack) in our implementation work. FIPS196 standard provides mutual authentication feature between CSPs. Role based access control (RABC) used for authorization support in SV2M. We have used XML Encryption/Signature for the confidentiality, integrity and source non-repudiation security features. We have enhanced the functionalities of Key Manager (KM) of CSP which support the storage of keys for SV2M system.This chapter first describe the related technologies which are used in our thesis work. Then we describes the different components of SV2M system with their functionalities. Complete work-flow and an activity diagram of SV2M is also discussed. Integration of SV2M with OpenStack components and its complete work-flow is also explained in detail.

4.1 Related Technologies of SV2M System

Implementation of the SV2M system for Cloud environment involves the use of PyXMLSec,OpenStack, M2Crypto and FIPS196 tools and specifications. We elaborate each one of them to help understand their functionality and usage in our thesis work.

a. **PyXMLSec** We selected XML encryption and XML digital signature in our thesis work. It is used to provide confidentiality, integrity and source non-repudiation security feature in SV2M. It has various features such as message level security, different keys for different document etc. PyXMLSec is an open source implementation of XML (Encryption/Digital Signature)feature in Python. We have used this library for XML encryption, Decryption and Digital signature functionalities in SV2M.

b. **OpenStack** OpenStack is open source Cloud platform which provides IaaS delivery model of Cloud computing The major components of OpenStack are Dashboard or graphical user interface (Horizon), Identity service (Keystone), Image Service (Glance), Storage infrastructure (Swift), Network service (Neutron), Volume service (Cinder) , Compute Infrastructure (Nova) [18, 38]. Horizon is the dashboard project which provides graphical interface to administrator and user to access, provision resources of Cloud. It is just one way to interact with OpenStack resources. Authentication and Authorization in OpenStack are provided by Keystone component. We have integrated our SV2M module in this Cloud platform.

c. **M2Crypto** M2Cypto is the popular M2Crypto library of Python which various security features such as symmetric algorithm (AES),Asymmetric algorithm (RSA) which is used for generation of Private/Public keys [39]. We have used this library in the one module of SV2M. It is used for generation of self signed certificate to use in SV2M module. M2Crypto is available as a as a fully supported package in the major distribution of Linux (Ubuntu,Fedora).

d. **FIPS196** It is Federal Information Processing Standard (FIPS)-196 for the mutual authentication between two entities. It is used in SV2M to achieve mutual authentication between CSPs. There is no Python implementation available for FIPS-196, therefore we contributed FIPS-196 implementation to open source community.

4.2 Proposed System Design

Our proposed solution provides the protection against active and passive attacks during migration process. The solution introduces strong security features such as mutual authentication between CSPs, authorization before initiation of migration operation, confidentiality, integrity, replay resistance and non-repudiation. The architecture of proposed solution is shown in Figure 4.1.

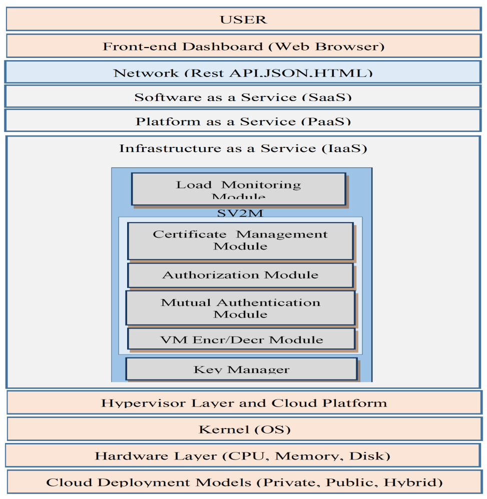

USER
Front-end Dashboard (Web Browser)
Network (Rest API,JSON,HTML)
Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)
Load Monitoring Module
SV2M
Certificate Management Module
Authorization Module
Mutual Authentication Module
VM Encr/Decr Module
Key Manager
Hypervisor Layer and Cloud Platform
Kernel (OS)
Hardware Layer (CPU, Memory, Disk)
Cloud Deployment Models (Private, Public, Hybrid)

Figure 4.1: Proposed System

Proposed secure VM migration (SV2M) system encompasses different components. Its core part is SV2M module which is further divided into sub modules such as i) certificate management module (CMM) ii) authorization module (AZM) iii) mutual authentication module (MAM) iv) encryption and decryption module (EDM). Other components are load monitoring module (LMM) for continuous monitoring of cloud resource and Key Manager (KM) for storing the keys which include encrypted disk image keys (EIK) and VM encryption keys (VMK). All these modules communicate with other components of Cloud platform. The detail functionalities of each module are described below.

4.2.1 Secure VM Migration (SV2M) Module

This module provides security while VM mobility occurs between CSP's. It consists of following four sub modules: Certificate management module (CMM), Authorization module (AZM), Mutual Authentication module (MAM) and VM encryption/decryption module (EDM). The detail description of sub modules are given below.

4.2.1.1 Certificate Management Module (CMM)

This module first generates RSA keys, stores the private key secret and then sends the certificate signing request (CSR) to Trusted Third Party (TTP) or Certification Authority (CA) for digital identity certificates. The digital certificate for the CSP is also store in the Cloud platform on certain path which is accessible to other modules. This X.509 v3 certificate is later used by the authentication module for entity authentication or mutual authentication of CSPs using FIPS-196 standard.

4.2.1.2 Authorization Module (AZM)

This module checks the authorization of current user/operator before initiating the process of secure VM migration. Secure VM migration can only be performed by those users which are allowed by the Cloud administrator. We are using Role based access control (RBAC) to restrict access to SV2M module. RBAC is a secure method of restricting operator access to authorized Cloud resources only.

4.2.1.3 Mutual Authentication Module (MAM)

Authenticity of sender and receiver is achieved using strong authentication mechanism. This module ensures that source and destination CSPs are authenticated by each other and ready to perform the VM migration. In this module, CSPs send X.509 certificates to each other and perform mutual authentication. We are using FIPS-196 standard to achieve mutual authenticity between CSPs before VM transfer. Random numbers from sender to receiver and Public Key Cryptography are used in FIPS-196 standard for mutual authentication [40].

4.2.1.4 VM Encryption and Decryption Module (EDM)

After successful mutual authentication between CSP's, the next step is encryption and digital signature of suspended/paused VM at sender CSP end. We are using XML Encryption and Signature in this module. First of all, XML signature of suspended VM is created and signed with private key (Priv_KA) of Cloud A. In the next step, encryption key (VMK) for suspended VM is retrieved from key manager and used for XML Encryption of VM using AES algorithm. In final step, both VMK and (EIK) are encrypted using the Public Key (Pub_KB) of Cloud Band transferred along XML encryption and signature of VM, as shown in Figure 4.2.

After the encrypted VM is received, the decryption module extracts the VMK and EIK using the private key (Priv_KB) of Cloud B. The EI key of received VM is stored in the Key Manager, whereas VMK is used to decrypt the migrated VM. For XML signature verification, ED module decrypt the

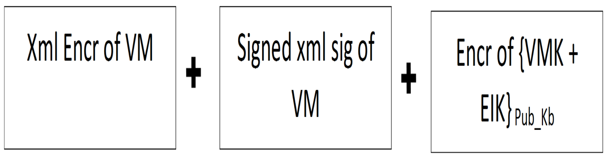


Figure 4.2: Encryption of message at sender side

signed XML signature of VM using Public key of Cloud A (Pub_KA) and compare it with newly created XML signature of decrypted VM as shown in Figure 4.3. After successful verification, suspended VM is resumed on receiver Cloud for providing services.

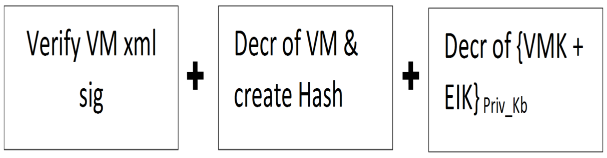


Figure 4.3: Decryption of message at receiver side

4.2.1.5 Key Manager (KM)

Key Manager provides key management such as generation, their secure storage, updation and deletion. CSP uses Key Managers for server side encryption such as Object Storage transparent encryption by Google and Amazon [41]. CSP also use it for storage of encrypted disk images keys (EIK) which are used to protect disk images in cloud repositories [42, 43, 44, 45]. It is also used for generation and storage of VM encryption keys (VMK) for ED module. In secure migration module, EIK and VMK are transferred to the receiver Cloud Provider along with encrypted VM. After successful resumption of VM, disk image key (EIK) is store on the key manager at receiver end.

4.2.1.6 Load Monitoring Module (LMM)

This module is used to monitor the Cloud resources which include the unused random access memory (RAM), storage and virtual CPU (vCPU) etc and linked to secure VM migration module (SV2M). It intimates the source Cloud (Cloud A) operator about the migration of VM due to unavailability of resources for new VM's. It also intimates the destination or receiver Cloud (Cloud B) that whether it can accept or reject the VM migration request based on available resources.

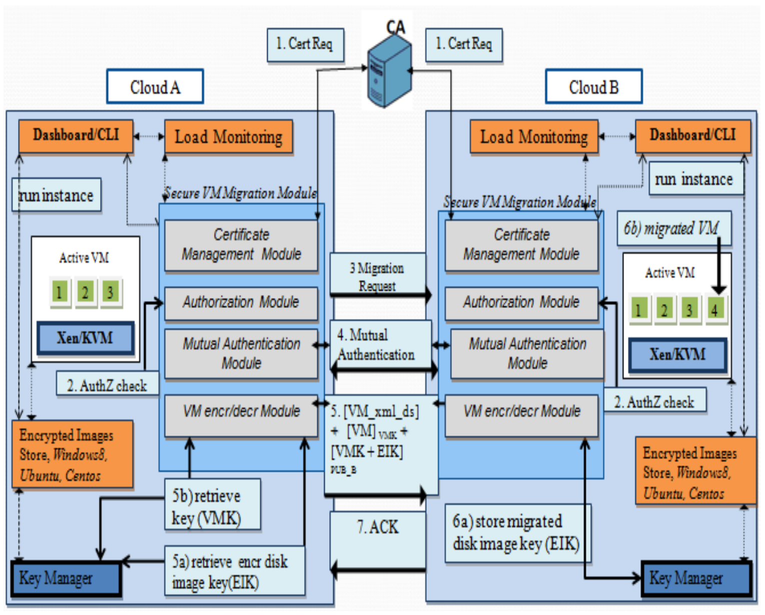

CA
1. Cert Req
1. Cert Req
Cloud A
Cloud B
Dashboard/CLI
Load Monitoring
Load Monitoring
Dashboard/CLI
run instance
run instance
Secure VM Migration Module
Secure VM Migration Module
Certificate Management Module
Certificate Management Module
Authorization Module
Authorization Module
Mutual Authentication Module
Mutual Authentication Module
VM encr/decr Module
VM encr/decr Module
Active VM
Active VM
Xen/KVM
Xen/KVM
6b) migrated VM
3 Migration Request
4. Mutual Authentication
5. [VM_xml_ds] + [VM] VMK + [VMK + EIK] PUB_B
7. ACK
2. AuthZ check
2. AuthZ check
Encrypted Images Store, Windows8, Ubuntu, Centos
Encrypted Images Store, Windows8, Ubuntu, Centos
5b) retrieve key (VMK)
5a) retrieve encr disk image key(EIK)
6a) store migrated disk image key (EIK)
Key Manager
Key Manager

Figure 4.4: Overall work-flow of SV2M module

Complete work-flow of proposed architecture is shown in Figure 4.4 and the description of each step is given below.

1. As a first step, source & destination CSPs request Trusted CA for certificates. These Certificates and their private keys are stored in Cloud Platform.

2. In this step, AZM checks that whether the Cloud operator can initiate request for VM migration operation or not.

3. After the successful authorization, CSP A initiates VM migration request toward CSP B. Request will be accepted by CSP if enough unused resources are available for migrated VM. Acceptance and rejection of migration request depends upon Load monitoring module because it intimates the SV2M module about the available resources. In this step, both CSP's mutually authenticate each other using FIPs196 entity authentication. Both parties share their X-509 certificate in this mechanism

4. Cloud providers have repositories of VM disk images which are encrypted to protect against offline attacks while at rest. When every time customer requests for VM, CSP first decrypt the disk image using key (EIK) which is stored in key manager and run it using hypervisors

such as Xen/KVM/VMware on Cloud. EI key is also migrated along with the suspended VM. Therefore Cloud Provider A first suspends running VM and retrieve corresponding EIK. XML signature of suspended VM are created and signed by the sender. XML encryption is also performed on VM using key VMK. Finally VMK & EIK are also encrypted using the public key of Cloud B and sent to Cloud B.

5. When Cloud B receives the encrypted VM with signatures and encrypted Keys (EIK & VMK), first it decrypts the encrypted keys using Private Key of Cloud B. EIK is stored in the key manager and VMK is used for decryption of encrypted VM. In parallel, Cloud B also decrypts the XML signatures of VM using Public key of Cloud A and compare it with XML signature of received VM. Migrated VM is resumed on Cloud B hypervisor after successful verification.

6. Cloud B acknowledges the Cloud A after successful execution of VM and it is removed from send CSP [15].

The activity diagram of complete work flow is shown in the Figure 4.5.

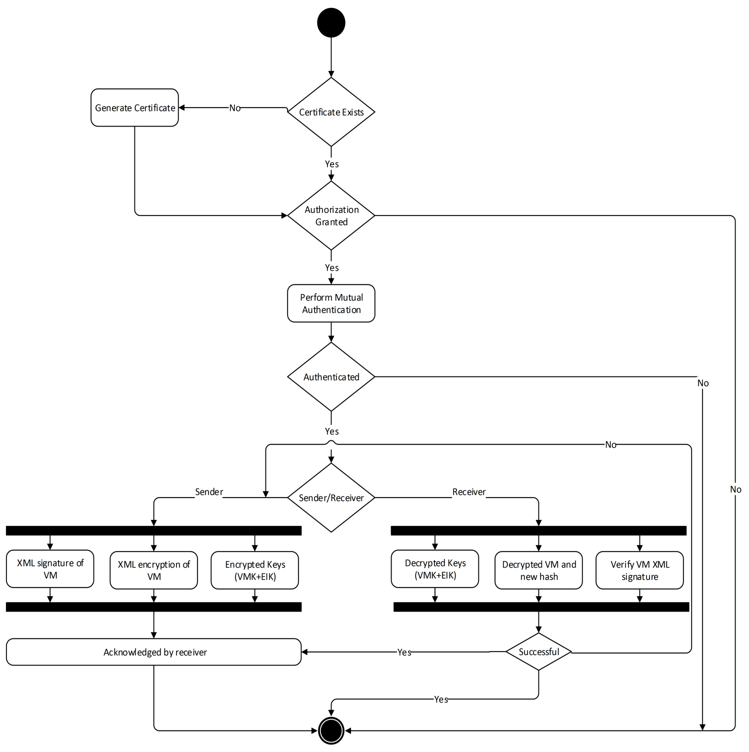


Figure 4.5: Activity diagram of SV2M

4.3 Integration of SV2M with OpenStack Platform

OpenStack is open source Cloud platform which provides IaaS delivery model of Cloud computing The major components of OpenStack are Dashboard or graphical user interface (Horizon), Identity service (Keystone), Image Service (Glance), Storage infrastructure (Swift), Network service (Neutron), Volume service (Cinder) , Compute Infrastructure (Nova) [18, 38]. Horizon is the dashboard project which provides graphical interface to administrator and user to access, provision resources of Cloud. It is just one way to interact with OpenStack resources. Authentication and Authorization in OpenStack are provided by Keystone component. Basically it provides identity, token and policy services in Cloud. It generates a token against each user or administrator which is used for authenticity of request during interactions of different components. Glance component of OpenStack is used for registering, listing, and retrieval of VM images. It also holds metadata and status information of the images. Glance provides an abstraction to multiple storage technologies such as object storage systems of OpenStack Swift project. Swift is the distributed storage system for managing of all types of storage such as archives, virtual machine images through RESTful HTTP API. There are multiple layers of redundancy, failover management and automatic replication; therefore loss of data in the node is reduced due to automatic recovery.

Furthermore, Neutron is focused on delivering networking as a service in OpenStack cloud environment, whereas, Cinder is block storage service and it manages the volumes in OpenStack. Compute Infrastructure is the virtual machine provisioning/management module of Cloud software and in OpenStack Cloud these functionalities are provided by Nova component. It is underlying cloud computing fabric controller and interacts with other components for provisioning and management of VM's. It manages all activities needed to support the life cycle (management such as creation, update and deletion) of instances within the OpenStack cloud. It uses libvirt virtualization library for hypervisor support such as VMware, Xen, KVM and Hyper-V in OpenStack. Nova uses sub components nova-compute and nova-scheduler for both cold and hot VM migration in OpenStack. In cold migration, nova first turns off the VM however it suspends or pause the VM in case of hot migration [18, 38].

4.3.1 SV2M at Sender Cloud

We assume that Cloud administrator or operator wants to transfer VM instance from one provider to another. First of all, CMM module generates certificate and stores in Cloud which is accessible in OpenStack dashboard. We are using Role based access control (RBAC) for authorization of migration request. Cloud administrator allows certain role to perform VM migration in OpenStack. We are achieving this by using the Keystone component which performs authorization in OpenStack Cloud. Once successfully authorized, VM migration request is sent to the destination Cloud by the sender SV2M module. Strong authentication of both sender and receiver is required so that attacker cannot masquerade in migration process. We are using FIPS196 which is based on Public Key Cryptography for mutual authentication phase of secure VM migration between CSPs.

After successful authentication, SV2M requests nova components for suspension of instance. Nova suspends the running instance of OpenStack cloud and gathers instance corresponding information or data such as volume, networking and imaged details from Cinder, Neutron and Glance components. It returns both VM and metadata to SV2M module for XML encryption and signatures. ED module of SV2M performs XML signature on VM and its metadata. After that, the SV2M requests KM for XML encryption key, VMK. We are using AES256 encryption algorithm with 256 bit key for VM encryption. Finally, SV2M also retrieves EIK from KM, encrypt both keys using receiver Public Key and sent along with XML signature and encrypted VM to the receiver cloud. Integration of SV2M with OpenStack at sender Cloud is shown in Figure 4.6.

4.3.2 SV2M at Receiver Cloud

SV2M module is also integrated with OpenStack at receiver side which performs the mutual authentication and accepts the encrypted message (which includes digital signature of VM, encrypted VM and keys). ED module decrypts the keys using Private Key of receiver and stores the EIK in the KM. It uses the VMK for XML Decryption and also calculates the hash of VM after decryption. ED module also decrypts the digital signature of VM and compares it with the newly created hash. After successful verification of hashes, SV2M module passes the VM and related metadata to nova component of OpenStack for resuming services in receiver Cloud.

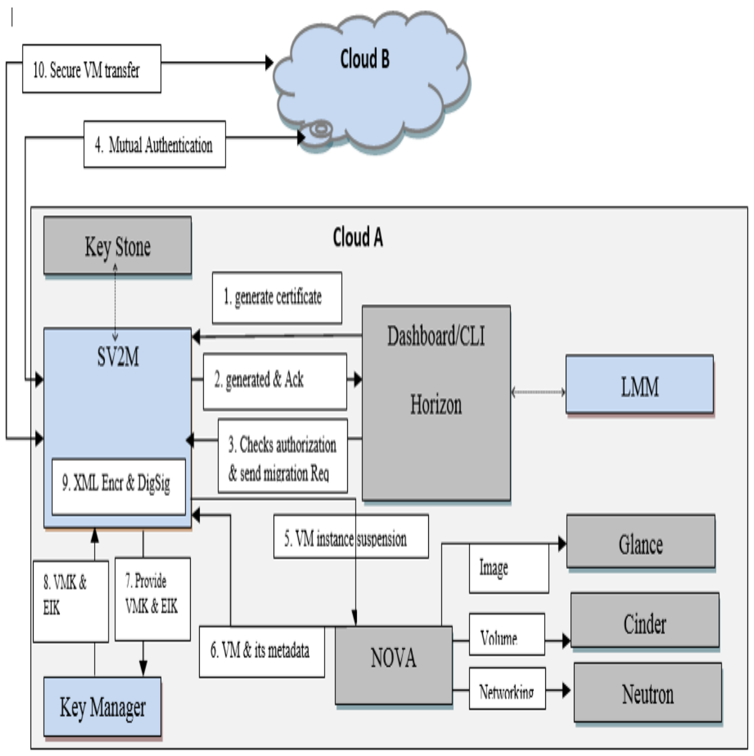
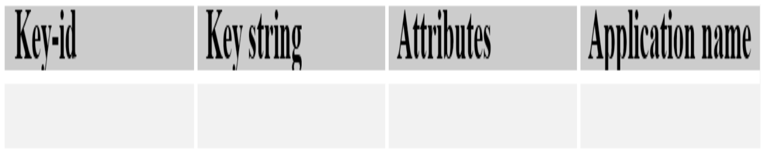


Figure 4.6: OpenStack and SV2M integration at sender end

4.4 Enhancements in Key Manager

Key management is very important component of any security solution. It provides the key generation, deletion and updation features. Normally key manager APIs support create, delete, update functions and different application or software use get and put functions for accessing the stored keys.We have stored different application keys in key manager table, which is shown in the Figure 4.7.



Key-id	Key string	Attributes	Application name

Figure 4.7: Representation of keys in Key Manager

We are using key manager in SV2M for storage of VMK and migration key. Enhanced Key Manager for SV2M is shown in the Figure 4.8. VM encryption keys are generated and stored in key manager which are used to provide confidentiality during its migration from one to anther Cloud. EIKs of VM are also stored in enhanced key manager after successful migration. EIK is used again on receiver Cloud to secure store images.

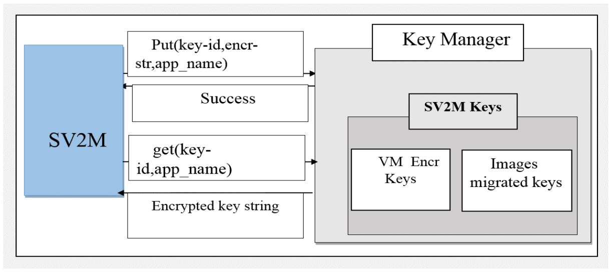


Figure 4.8: Enhanced OpenStack Key Manager for SV2M

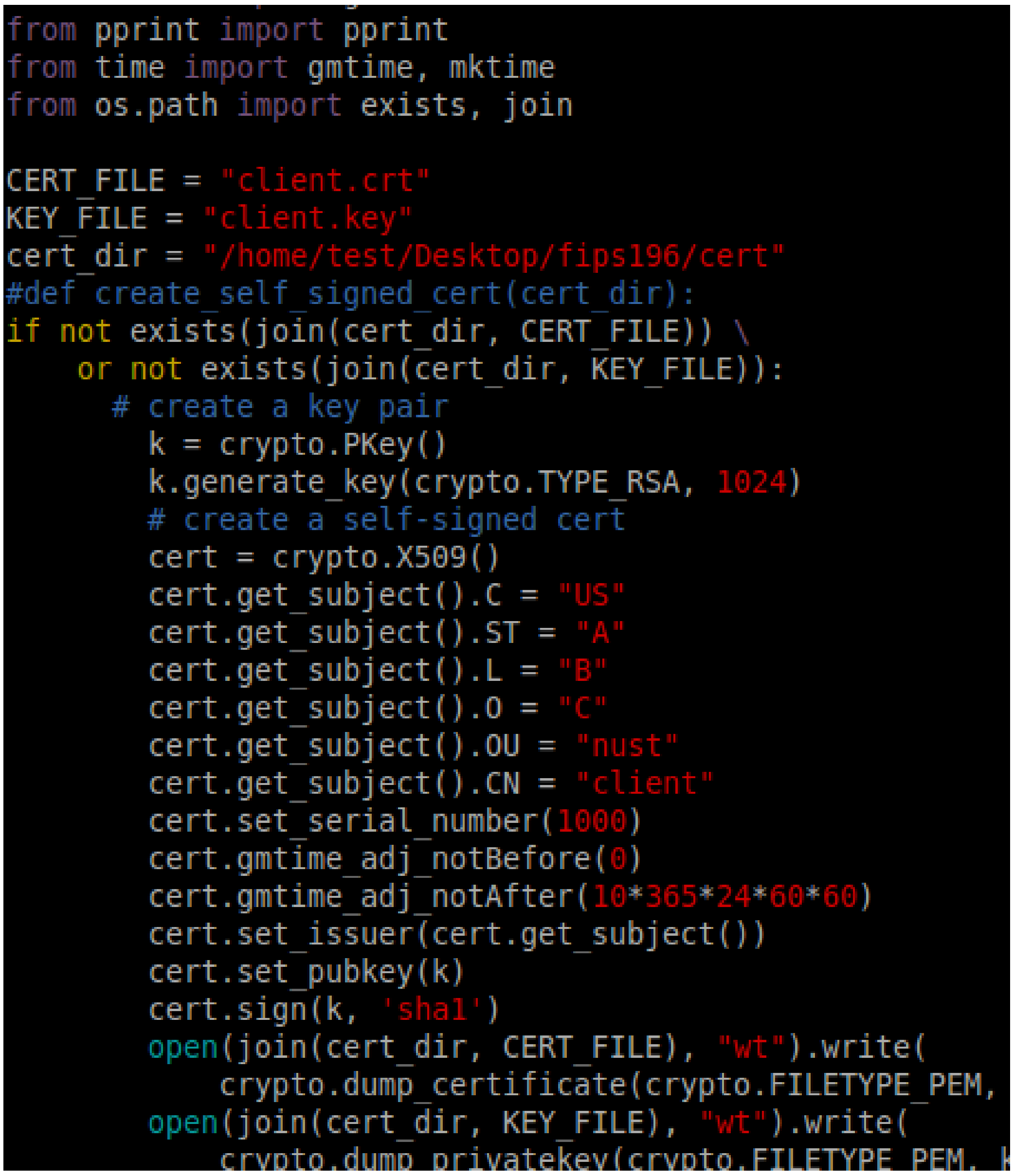
4.5 Implementation

This sections discusses the details about the implementation work on SV2M system. We have integrated SV2M framework with OpenStack and it interacts with Nova,Glance,Quantum modules. SV2M system is developed using Python and bash scripting languages. Implementation details of modules is given below.

4.5.1 Certificate Generation and Mutual Authentication

The detail about the environment setup for SV2M system is given in Appendix A. First of all, Python script generates the certificates for the source Cloud (Client) & Destination Cloud (Server). Certificate generation code for client/server is shown in Figure 4.9.

These certificate are further used in Mutual Authentication Module of SV2M. MAM is implemented in Client/Server model in SV2M. Sender CSP is the client and receiving CSP is the Server which listens from the incoming request from sender for VM migration. Code snippet of client and server is shown in Figure 4.10 & Figure 4.11

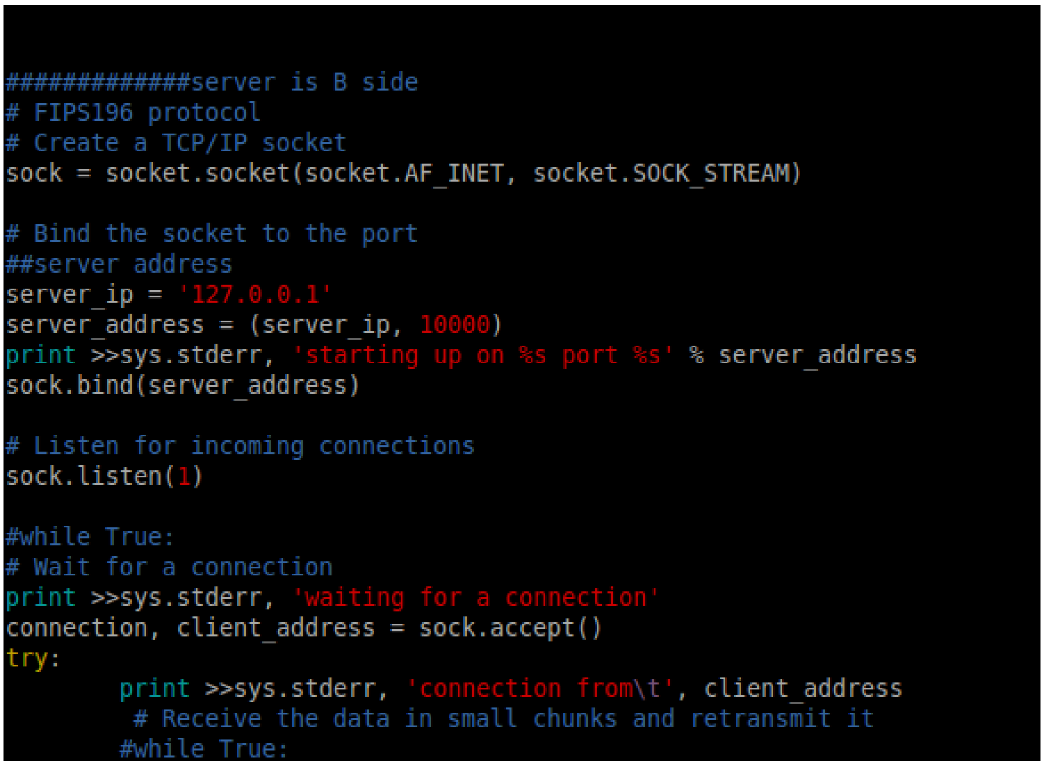


```
from pprint import pprint
from time import gmtime, mktime
from os.path import exists, join

CERT_FILE = "client.crt"
KEY_FILE = "client.key"
cert_dir = "/home/test/Desktop/fips196/cert"
#def create_self_signed_cert(cert_dir):
if not exists(join(cert_dir, CERT_FILE)) \
    or not exists(join(cert_dir, KEY_FILE)):
      # create a key pair
        k = crypto.PKey()
        k.generate_key(crypto.TYPE_RSA, 1024)
        # create a self-signed cert
        cert = crypto.X509()
        cert.get_subject().C = "US"
        cert.get_subject().ST = "A"
        cert.get_subject().L = "B"
        cert.get_subject().O = "C"
        cert.get_subject().OU = "nust"
        cert.get_subject().CN = "client"
        cert.set_serial_number(1000)
        cert.gmtime_adj_notBefore(0)
        cert.gmtime_adj_notAfter(10*365*24*60*60)
        cert.set_issuer(cert.get_subject())
        cert.set_pubkey(k)
        cert.sign(k, 'sha1')
        open(join(cert_dir, CERT_FILE), "wt").write(
            crypto.dump_certificate(crypto.FILETYPE_PEM,
        open(join(cert_dir, KEY_FILE), "wt").write(
            crypto.dump_privatekey(crypto.FILETYPE_PEM, k
```

Figure 4.9: Certificate Generation Code Snippet

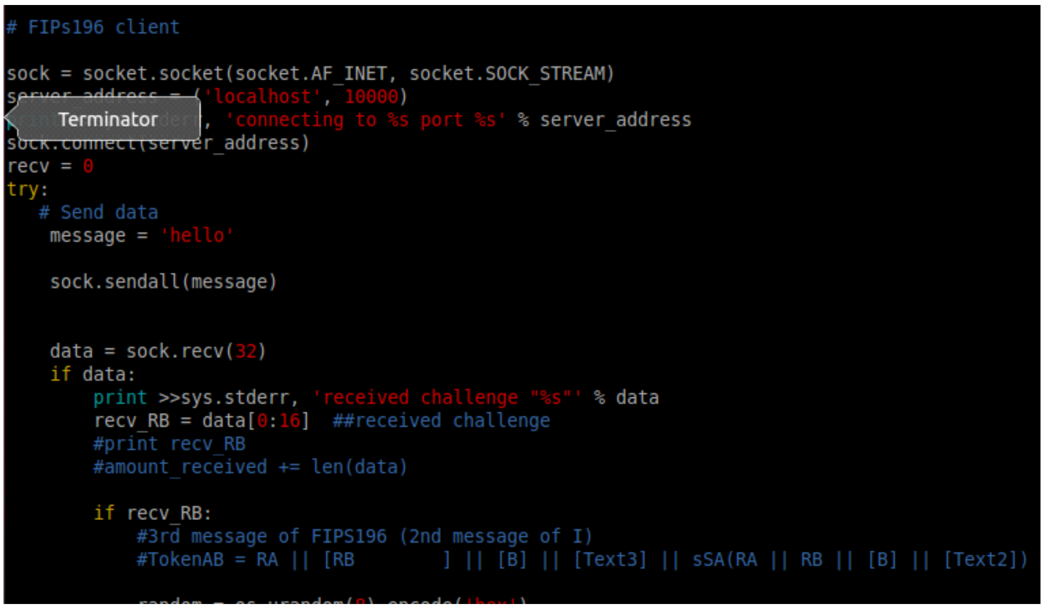
4.5.2 XML encryption & Digital Signature

we have a bash script to call python scripts which provides the encryption, decryption, random key generation and digital signature functionalities in SV2M. Suspension and resumption of VM are also provided by this bash script (scr_vmmigration.sh). Code snippet of SV2M main script is shown in Figure 4.12.



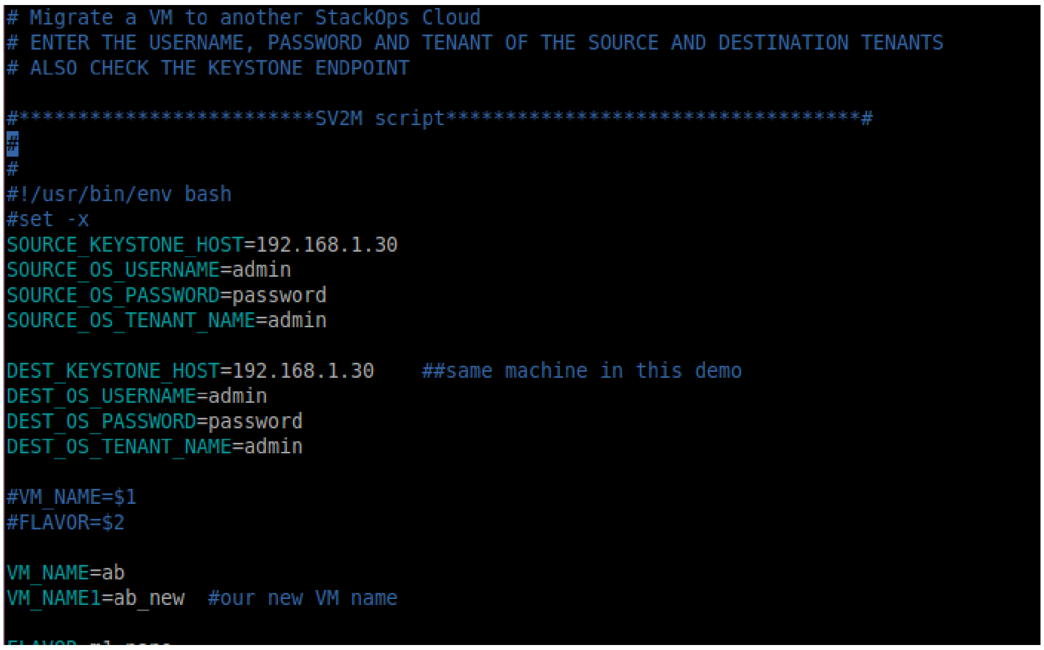
```
#############server is B side
# FIPS196 protocol
# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the port
##server address
server_ip = '127.0.0.1'
server_address = (server_ip, 10000)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address)

# Listen for incoming connections
sock.listen(1)

#while True:
# Wait for a connection
print >>sys.stderr, 'waiting for a connection'
connection, client_address = sock.accept()
try:
        print >>sys.stderr, 'connection from\t', client_address
         # Receive the data in small chunks and retransmit it
        #while True:
```

Figure 4.10: Server Cloud Code for MAM



```
# FIPs196 client

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_address = ('localhost', 10000)
[illegible], 'connecting to %s port %s' % server_address
sock.connect(server_address)
recv = 0
try:
   # Send data
    message = 'hello'

    sock.sendall(message)


    data = sock.recv(32)
    if data:
        print >>sys.stderr, 'received challenge "%s"' % data
        recv_RB = data[0:16]  ##received challenge
        #print recv_RB
        #amount_received += len(data)

        if recv_RB:
            #3rd message of FIPS196 (2nd message of I)
            #TokenAB = RA || [RB        ] || [B] || [Text3] || sSA(RA || RB || [B] || [Text2])
```

Figure 4.11: Client Cloud Code for MAM



```
# Migrate a VM to another StackOps Cloud
# ENTER THE USERNAME, PASSWORD AND TENANT OF THE SOURCE AND DESTINATION TENANTS
# ALSO CHECK THE KEYSTONE ENDPOINT

#*************************SV2M script***********************************#
#
#
#!/usr/bin/env bash
#set -x
SOURCE_KEYSTONE_HOST=192.168.1.30
SOURCE_OS_USERNAME=admin
SOURCE_OS_PASSWORD=password
SOURCE_OS_TENANT_NAME=admin

DEST_KEYSTONE_HOST=192.168.1.30    ##same machine in this demo
DEST_OS_USERNAME=admin
DEST_OS_PASSWORD=password
DEST_OS_TENANT_NAME=admin

#VM_NAME=$1
#FLAVOR=$2

VM_NAME=ab
VM_NAME1=ab_new  #our new VM name
```

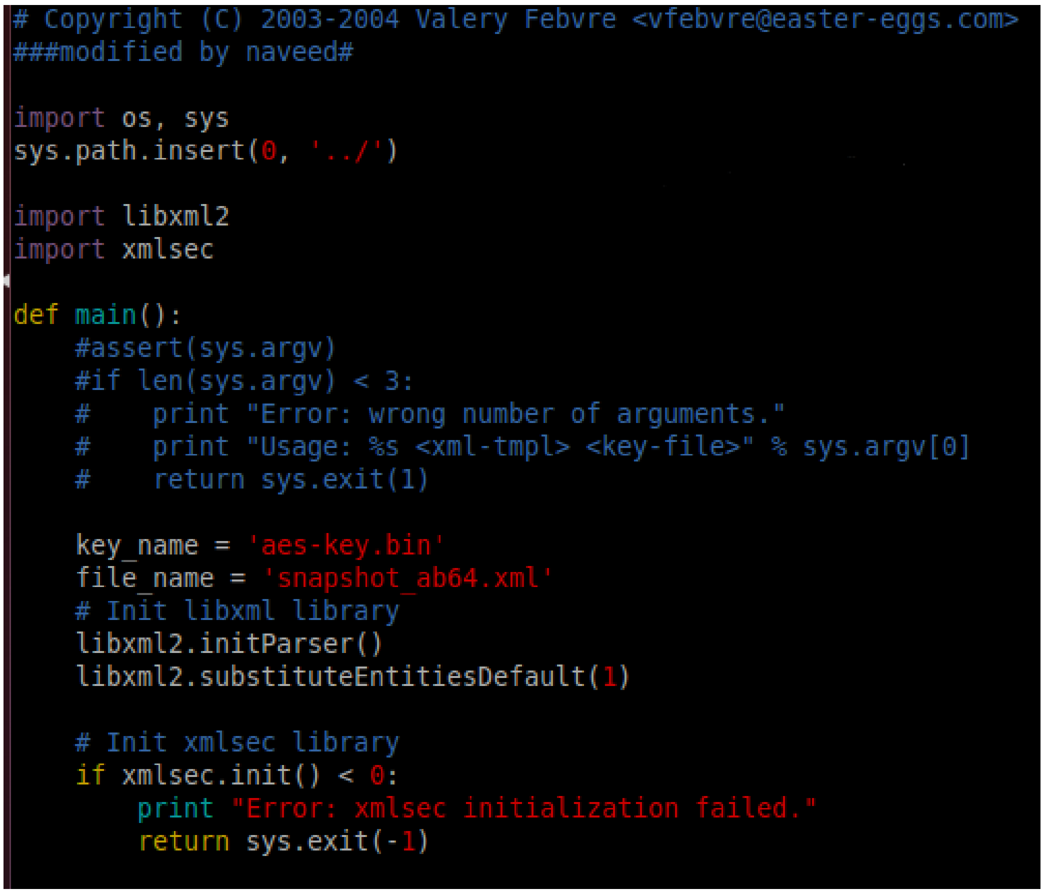
Figure 4.12: SV2M System

PyXMLSec is a set of Python bindings for the XML Security Library. XML Security Library is implemented in C programming language and it is based on LibXML2. The current version of library supports following XML security standards:

1. XML Signature
2. XML Encryption
3. Canonical XML
4. Exclusive Canonical XML

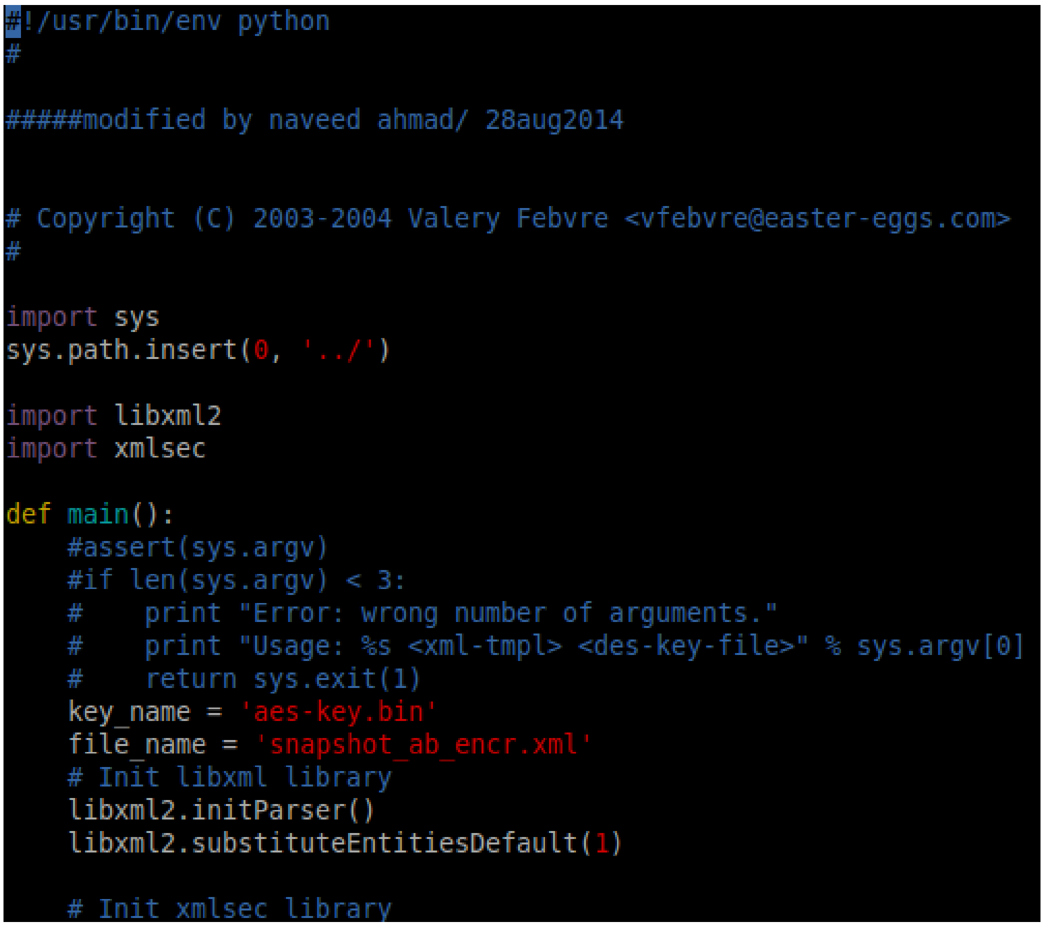
Snippets of encryption & decryption code which uses PyXMLSec library are show in Figure 4.13 & Figure 4.14.

PyXMLSec also supports XML digital signature and we have used it for integrity security feature in SV2M system. Code snippets for Digital Signature and its Verification are shown in Figure 4.15 & Figure 4.16.



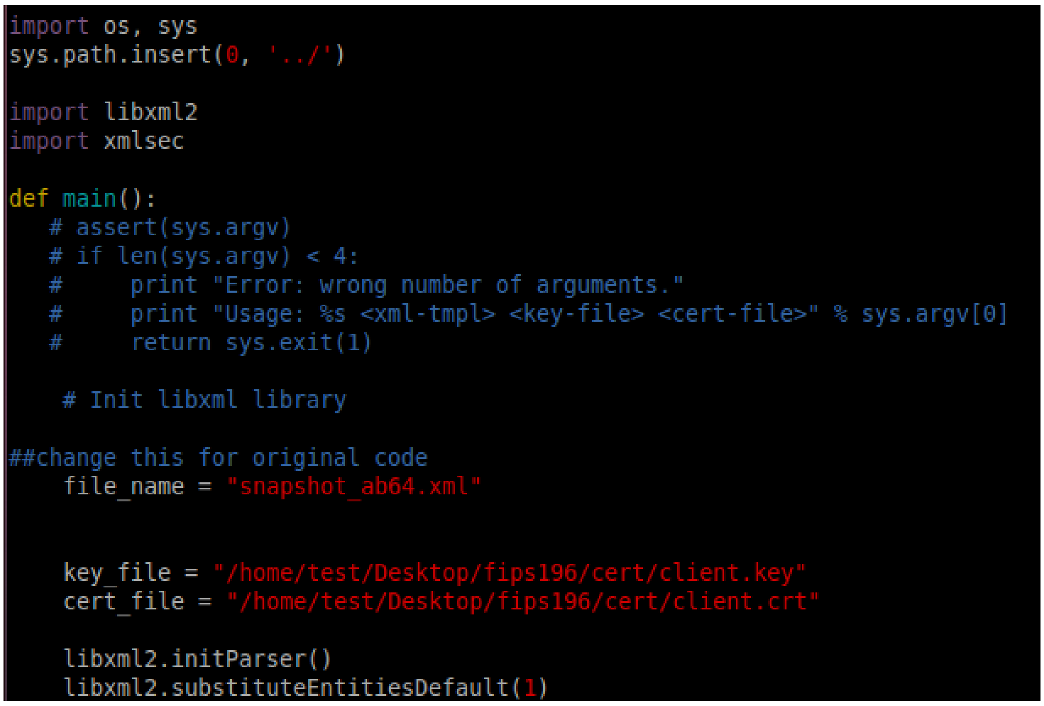
```
# Copyright (C) 2003-2004 Valery Febvre <vfebvre@easter-eggs.com>
###modified by naveed#

import os, sys
sys.path.insert(0, '../')

import libxml2
import xmlsec

def main():
    #assert(sys.argv)
    #if len(sys.argv) < 3:
    #    print "Error: wrong number of arguments."
    #    print "Usage: %s <xml-tmpl> <key-file>" % sys.argv[0]
    #    return sys.exit(1)

    key_name = 'aes-key.bin'
    file_name = 'snapshot_ab64.xml'
    # Init libxml library
    libxml2.initParser()
    libxml2.substituteEntitiesDefault(1)

    # Init xmlsec library
    if xmlsec.init() < 0:
        print "Error: xmlsec initialization failed."
        return sys.exit(-1)
```

Figure 4.13: Code for XML Encryption



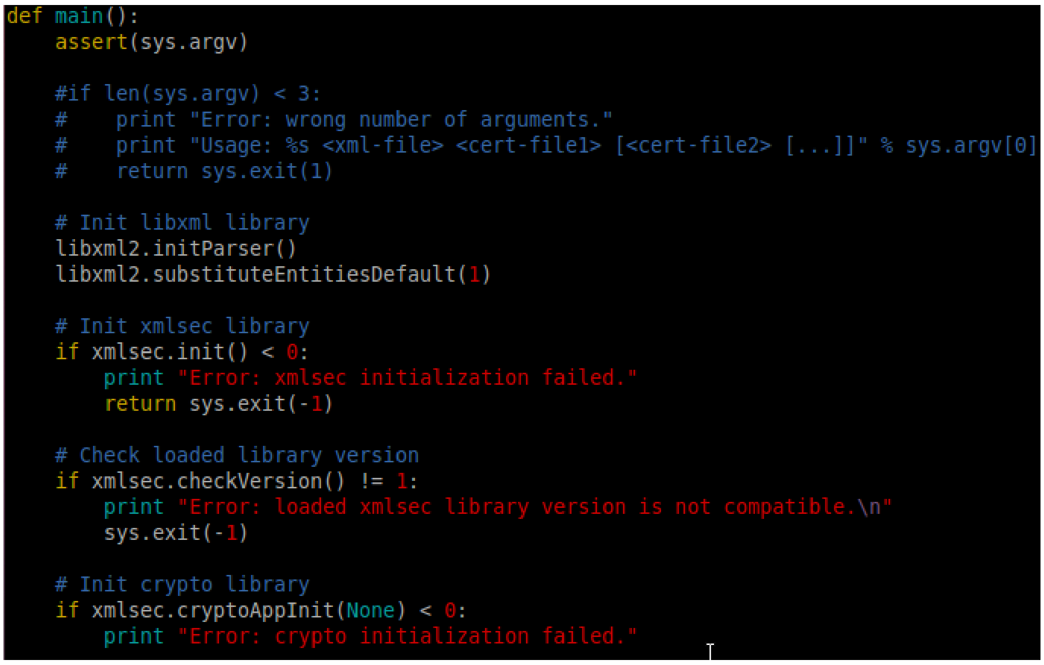
```
#!/usr/bin/env python
#

#####modified by naveed ahmad/ 28aug2014


# Copyright (C) 2003-2004 Valery Febvre <vfebvre@easter-eggs.com>
#

import sys
sys.path.insert(0, '../')

import libxml2
import xmlsec

def main():
    #assert(sys.argv)
    #if len(sys.argv) < 3:
    #    print "Error: wrong number of arguments."
    #    print "Usage: %s <xml-tmpl> <des-key-file>" % sys.argv[0]
    #    return sys.exit(1)
    key_name = 'aes-key.bin'
    file_name = 'snapshot_ab_encr.xml'
    # Init libxml library
    libxml2.initParser()
    libxml2.substituteEntitiesDefault(1)

    # Init xmlsec library
```

Figure 4.14: Code for XML Decryption



```
import os, sys
sys.path.insert(0, '../')

import libxml2
import xmlsec

def main():
   # assert(sys.argv)
   # if len(sys.argv) < 4:
   #     print "Error: wrong number of arguments."
   #     print "Usage: %s <xml-tmpl> <key-file> <cert-file>" % sys.argv[0]
   #     return sys.exit(1)

    # Init libxml library

##change this for original code
    file_name = "snapshot_ab64.xml"


    key_file = "/home/test/Desktop/fips196/cert/client.key"
    cert_file = "/home/test/Desktop/fips196/cert/client.crt"

    libxml2.initParser()
    libxml2.substituteEntitiesDefault(1)
```

Figure 4.15: Code for XML Digital Signature



```
def main():
    assert(sys.argv)

    #if len(sys.argv) < 3:
    #    print "Error: wrong number of arguments."
    #    print "Usage: %s <xml-file> <cert-file1> [<cert-file2> [...]]" % sys.argv[0]
    #    return sys.exit(1)

    # Init libxml library
    libxml2.initParser()
    libxml2.substituteEntitiesDefault(1)

    # Init xmlsec library
    if xmlsec.init() < 0:
        print "Error: xmlsec initialization failed."
        return sys.exit(-1)

    # Check loaded library version
    if xmlsec.checkVersion() != 1:
        print "Error: loaded xmlsec library version is not compatible.\n"
        sys.exit(-1)

    # Init crypto library
    if xmlsec.cryptoAppInit(None) < 0:
        print "Error: crypto initialization failed."
```

Figure 4.16: Code for XML Signature Verification in SV2M

Chapter 5

Security and Performance Evaluation

This chapter explains the results of evaluation that we have done for our proposed solution. We have designed a holistic solution named it SV2M for the security of VM migration process which fulfills the identified security requirements. We also used AVISPA, a well-known security protocol verification tool to validate our system against identified vulnerabilities and threats. Performance or SV2M is tested using time utility which is available in Linux

5.1 Security Requirements for SV2M system

This section presents the those essential security requirements which are supported by SV2M solution in Cloud environment . The detail of each requirement is given below.

1. ***Integrity Verification of Platform:***

   The destination platform cryptographically identifies itself to source for trust establishment.

2. ***Mutual Authentication:***

   Attacker can launch MITM attack using technique such as route hijacking or ARP poisoning in the migration process. In order to avoid MITM attacks on live VM migration, source and destination platforms must mutually authenticate each other.

3. ***Authorization (Access control policies):***

Appropriate access control policies must be provided to secure the live VM migration process. An unauthorized user/role may launch VM initiate, migration operation. Unauthorized activities can be prevented using access control list (ACL's) in hypervisor.

4. ***Confidentiality and Integrity of VM during migration:***

   An encrypted channel must be established so that an attacker can not get any information from VM contents and modification of contents can be properly detected. This will help to avoid active attacks such as memory manipulation on live migration and passive attacks such as leakage of sensitive information.

5. ***Replay Resistance:***

   Attacker can capture traffic and replay it latter to get authenticated in VM migration process. Therefore live VM migration process should be replay resistant. Nonce's can be used to prevent replay attack in migration.

6. ***Source Non-Repudiation:***

   Source host cannot deny from VM migration operation This feature can be achieved by using Public key certificate.

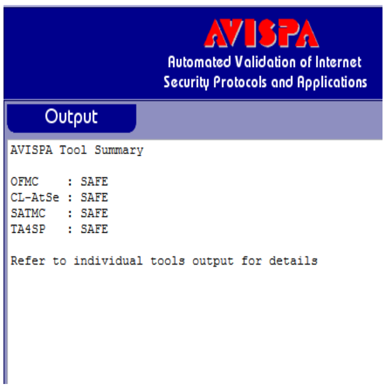
5.2 Protocol Validation Using AVISPA Tool

AVISPA is web based automated tool for the validation of security protocols and application. It uses a formal language known as High Level Protocol Specification Language (HLPSL) for protocol specification and their security properties/goals and integrates different back-ends such as SATMC, TA4SP ,CL-AtSe and OFMC for implementation of range of analysis techniques [46].The back-ends techniques are used to detect attacks on the given protocol or application and verification as well against infinite number of sessions. It is assumed in analysis that the protocol meets cryptography and also network over message exchanged is controlled by Dolev-Yao intruder as an active intruder. On the fly Model Checker analysis technique implements both bounded session verification and protocol falsification. OFMC helps in finding known attacks on protocol and discovers new ones. The Constraint Logic based attack searcher implements the deduction rules to execute protocols automatically under the attack of an intruder Dolev-Yao that possess the deduction capabilities. It is consider in the model checker which is based on SAT that the presented protocol has finite number of sessions during the message exchange under active intruder Dolev-Yao. Tree Automata based Protocol Analyzer back-end provides protocol verification using approximation method based on tree automata. An approximation helps to determine the result of the analysis; therefore in the case of secrecy properties if the result obtained is an over-approximation, then protocol is safe [47].

AVISPA analyzed the protocol against security goals such as secrecy of key, weak/strong authentication etc. We have analysed the secure migration protocol against our identified security requirements including:

1. Strong authentication (G1, G5)
2. Non-repudiation (G18)
3. Secrecy (G12), integrity (G2)
4. Reply protection (G3) [46, 47].

The result of back end analysis techniques against above mention security properties is shown in Figure 5.1. The output summary of AVISPA indicates that a secure VM migration protocol is safe under analysis of OFMC, CL-AtSe, and SATMC and TA4SP back-ends and no attack on found on it.


AVISPA
Automated Validation of Internet Security Protocols and Applications
Output

```
AVISPA Tool Summary

OFMC    : SAFE
CL-AtSe : SAFE
SATMC   : SAFE
TA4SP   : SAFE

Refer to individual tools output for details
```

Figure 5.1: Result summary of AVISPA

The hplsr script of AVISPA, stating different roles for the SV2M system and as well as for attacker with desirable security goals of protocol are given below.

AVISPA HLPSL Program

```
% Secure VM migration module
role client_cloud(A, B : agent,
           H : hash_func,
           Ka, Kt: public_key,  %% Kt is the public key of a TTP
           SND, RCV: channel (dy))
played_by A
def=
   local Ra : text,
         Rb: text,
         State: nat,
         Kb: public_key,
         Kvmk,K_eik : symmetric_key,
         VM : text
       const vm_encr_key : protocol_id
   init  State := 0

   transition

   1.  State = 0
       /\ RCV(start)
       =|>
       State' := 2
       /\ SND(A)

     2.  State = 2
       /\ RCV(Rb') =|>
       State' := 3
       /\ Ra' := new()
       /\ SND (Ra'.Rb'.{H(Ra'.Rb')}_(inv(Ka)).{A.Ka}_(inv(Kt)))
       /\ witness(A,B,bob_alice_num,H(Ra'.Rb'))

   4.  State = 3
     /\ RCV (Rb.Ra.{H(Rb.Ra)}_(inv(Kb)).{B.Kb}_(inv(Kt)))
          =|>
       State' := 5
      /\ request(B,A,alice_bob_num,H(Ra.Rb))
      /\ SND (({VM}_Kvmk).({H(VM)}_(inv(Ka))).{(Kvmk.K_eik)}_Kb)
      /\ secret(Kvmk,vm_encr_key,{A,B})

end role
role server_cloud(A, B : agent,
         H : hash_func,
         Kb, Kt: public_key,
         SND, RCV: channel (dy))
played_by B
def=

   local Rb, Ra: text,
         State: nat,
         Ka: public_key,
         Kvmk,K_eik : symmetric_key,
         VM : text
   const vm_encr_key : protocol_id
   init  State := 1

   transition

   1.  State = 1
       /\ RCV(A)   =|>
       State’ := 3
       /\ Rb’ := new()
       /\ SND(Rb’)
   2.  State = 3
       /\ RCV(Ra’.Rb.{H(Ra’.Rb)}_(inv(Ka)).{A.Ka}_(inv(Kt)))
       =|>
       State’ := 5
       /\ SND(Rb.Ra’.{H(Rb.Ra’)}_(inv(Kb)).{B.Kb}_(inv(Kt)))
       /\ witness(B,A,alice_bob_num,H(Rb.Ra’))
    3.  State = 5
          /\ RCV (({VM’}_Kvmk’).({H(VM’)}_(inv(Ka))).{(Kvmk’.K_eik’)
           =|>
       State’ := 7
       /\ secret(Kvmk’,vm_encr_key,{A,B})
       /\ request(A,B,bob_alice_num,H(Rb.Ra))
end role

role session(A,B: agent,
             Ka, Kb, Kt: public_key,
             H : hash_func)
def=

   local  SA, SB, RA, RB: channel (dy)

   composition
```

```
                client_cloud(A,B,H,Ka,Kt,SA,RA)
           /\   server_cloud(A,B,H,Kb,Kt,SB,RB)

end role

role environment()
def=
   const alice_bob_num,bob_alice_num  : protocol_id,
           h       : hash_func,
         a, b           : agent,
         ka, kb, ki, kt : public_key

   intruder_knowledge = { a, b, ka,{a.ka}_(inv(kt)),kb,{b.kb}_(inv(k
                           {i.ki}_(inv(kt)) }

   composition
        session(a,b,ka,kb,kt,h)
    /\  session(a,i,ka,ki,kt,h)
    /\  session(i,b,ki,kb,kt,h)

end role

goal
  secrecy_of vm_encr_key  % Addresses  G12
  %G5
   authentication_on alice_bob_num  % Addresses G1, G2, G3 ,G18
  authentication_on bob_alice_num  % Addresses G1, G2, G3 , G18

end goal

environment()
```

5.3 Deployment Setup for Performance Evaluation

We setup a test scenario in lab environment for securing VM migration from one cloud to another using OpenStack Cloud. We are using Linux distribution Ubuntu 14.04 LTS on Dell Power Edge R420 server which has Intel

Xeon processor (2.20GHz,10m cache), 450GB storage and 32GB RAM specifications. We have deployed OpenStack Cloud using devstack development environment with SV2M on these servers. We assume that both CSPs are connected to each other with one Gig link for migration traffic. OpenStack provides dashboard and cli to access Cloud resources. We have created new instance on sender Cloud using dashboard which has 512MB RAM,1GB storage and 1VCPU (nano flavor type) resources.

We assume that sender Cloud provider faced some resources issue and wants to transfer running instance to remote CSP. SV2M which is integrated with OpenStack, uses RBAC to check authority of migration operator and then perform mutual authentication between providers. Once mutual authentication is achieved, ED module of SV2M performs XML signature, XML encryption on VM and keys, and sent to the receiver devstack based OpenStack Cloud. Same ED module at receiver Cloud decrypts the received messages and launches the instance for providing services.

5.3.1 Performance Evaluation

In this section we perform the evaluation of SV2M which is integrated with OpenStack Cloud Platform. We have already described about the specification of our servers in the previous section. Security evaluation or verification of the system already perform using AVISPA in section 6. In this section we consider execution time as a parameters for the performance evaluation of SV2M solution of migrating instance from one OpenStack Cloud to another Cloud. The detail of instance running on sender side Cloud is given in Table 5.1.

Instance	Detail
Name	test
Flavour	m1.nano
Image Name	cirros-0.3.0-i386-disk.img
Resources	RAM 512 MB , Disk 1GB , VCPUs 1

Table 5.1: Detail of instance on sender OpenStack Cloud

In our implementation, MAM and EDM of SV2M are used for secure transfer of VM between sender/receiver Cloud. However, there is always trade-off between security and performance, Therefore we are using time utility which is available in Linux OS as a benchmarking tool. It is used to track execution time of application or program. Time command gives timing statistics about the program run on the terminal. These statistics consist of (i) the elapsed real time between request and termination (ii) the user CPU time and (iii) the system CPU time. Real time shows the execution time, user and sys tracks the CPU processing time [48]. Performance result of insecure and SV2M are shown in the Figure 5.2.

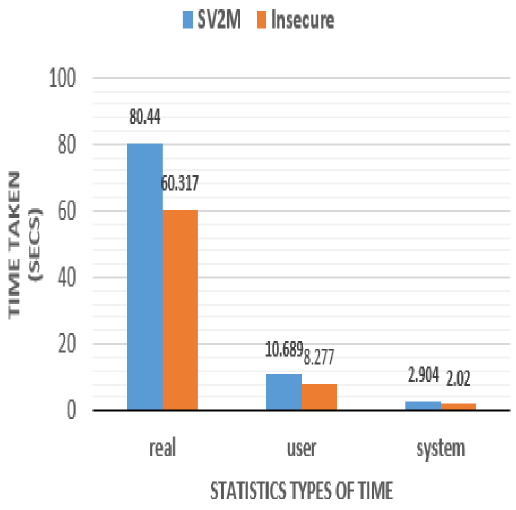


Figure 5.2: Secure vs Insecure migration performance evaluation

As shown in the above graph, blue bar is representing the time taken by SV2M and insecure migration is represented by the orange bar. It indicates that SV2M takes more time as compare to insecure migration. It will increase the downtime but security is our main focus during VM migration. Secure VM migration introduces overhead due to mutual authentication, encryption and signature security features.

Chapter 6

Conclusion & Future Directions

This chapter explains the summary of our thesis research working findings , its conclusion and future research directions . In this research work, we highlights the an important problems in the domain of virtualization security in Cloud environment that are still needed to be addressed. In the first, we summarize our research contributions that we made in during thesis work and then, a conclusion of our findings in the domain of secure VM migration Systems is presented. In the end, this chapter presents some of the future potential research areas linked to virutalization security in Cloud.

6.1 Thesis Contributions: A Summary

We have carried out research into two phases including 1) **Identification of security issue in Cloud due to usage of virtualization technology** and 2) **identification of security requirements for secure VM migration and design a solution which fulfill identified security requirements**. The summary of our research contribution is given below:

1. In our first contribution, we have conducted an in-depth literature survey to explore the security issues of virtualization technology in Cloud environment. During the literature review, we identified that VM migration is useful feature of virtualization and it is used in traditional data center. However due to security issues, it is not used in Cloud environment by the CSPs. So we focused to identify the security requirements for secure VM migration in Cloud environment. As a result, we propose an assessment criteria for the evaluation of secure VM migration systems for Cloud; comprising of potential security features which must be supported by secure migration system of Cloud.

2. In our next and final contribution, we have designed and implemented a prototype of secure VM migration (SV2M) system for Cloud environments that ensure the strong security features such as mutual authentication, confidentiality,Integrity,authorization, source non-repudiation and replay protection. We also proposed few enhancement in Key manager of CSP which provides storage of keys for SV2M system. Finally, we have integrated secure migration module for VM into the OpenStack Cloud platform. In the end,we have analyzed the security of SV2M using AVISPA verification tool.Performance of SV2M is tested using time utility of Linux which measures different parameters of program during its execution.

6.2 Conclusion

In this era of hacking and state sponsored attacks on services provider such as exposed by Edward Sonwden, security of VM migration in Cloud environment has emerged as one of the hot issue, therefore it has received significant attention form the research community and CSP around the globe. So far, many solutions have been proposed for secure VM migration in Cloud however, most of those systems either required many changes in current infrastructure of Cloud or they lacks in providing all security requirements during VM migration.

In this research work, first we have investigated the vulnerabilities and threats on the VM migration. Furthermore, we have defined security requirements for the detailed analysis of existing solutions and explored their limitations. There is no complete solution for VM migration which fulfills the identified security requirements. Furthermore, VM migration is not supported in vTPM based solutions.

After identification of security issue of VM migration, we have proposed holistic solution for secure VM migration which provides strong security features such as mutual authentication between Cloud Providers, authorization of migration operation, confidentiality and integrity of VM content, replay resistance and non-repudiation of source Cloud. Our solution requires least changes in existing Cloud infrastructures because it is not reliant on any hardware chip (TPM). We have integrated SV2M with OpenStack Cloud and also deployed it on Dell server. We have done the performance and security evaluation of our work, however in future, we will evaluate its performance with other existing solutions and decrease the encryption/ decryption time of VM migration module.

6.3 Future Research Directions

Security of Virtualization technology for Cloud environment is a broad domain which involves many potential area of research work by the research community and CSP such as Google,IBM,Rackspace etc.

1. Security aspects of receiver Cloud/destination Cloud after successful migration is a potential research area. Research on security challenges after migration of a malicious VM near highly critical machine in the receiver Cloud

2. Cloud providers are also using snapshots for instance migration between private and public Cloud, therefore security in snapshot migration is our future research work goal.

3. Like the security of disk images, security of instance snapshots is also potential research area because it is required in IaaS delivery model.

Appendix A

SV2M system deployment and Installation Manual

Preface
This SV2M system deployment manual is aimed at developers ,system administrators and technologists eager to understand and deploy a Secure VM Migration System in Cloud environment. This manual intends to help the organizations which are using their Private Cloud for IaaS . SV2M is designed to provide strong security during Cloud bursting feature (transferring a VM from Private to Public Cloud provider). This manual describes the procedure for the deployment of SV2M in OpenStack Cloud platform installed on Ubuntu 12.04LTS OS. Instruction in the manual describe how to install openstack, deploy SV2M and execute its different module.

Target Audience
The purpose of this manual is to create a guide for beginners who are new in virtualization and Cloud computing domain. However very basic level knowledge of linux and python is required to execute SV2M scripts in OpenStack Cloud.

A.1 Installation of OpenStack development version

We required development version of OpenStack known as DevStack for the deployment of SV2M system. It is very easy to deploy development environment of OpenStack using devstack. DevStack is a bash shell script which is used to build complete OpenStack development environments and is available at http://devstack.org/.

It is recommended that to install DevStack on Virtual Machine (VM) instead on physical system with any Linux distribution as OS . We used following configuration for Devstack installation.

- HOST OS on our Dell machine is: Ubuntu 12.04 LTS &
- GUEST OS installed on VM with static IP address (192.168.1.30) assigned: Ubuntu 12.04 LTS

Devstack script is hosted on Git Hub so git package is required to locally store it on guest VM. It can be installed using following commands on Ubuntu:

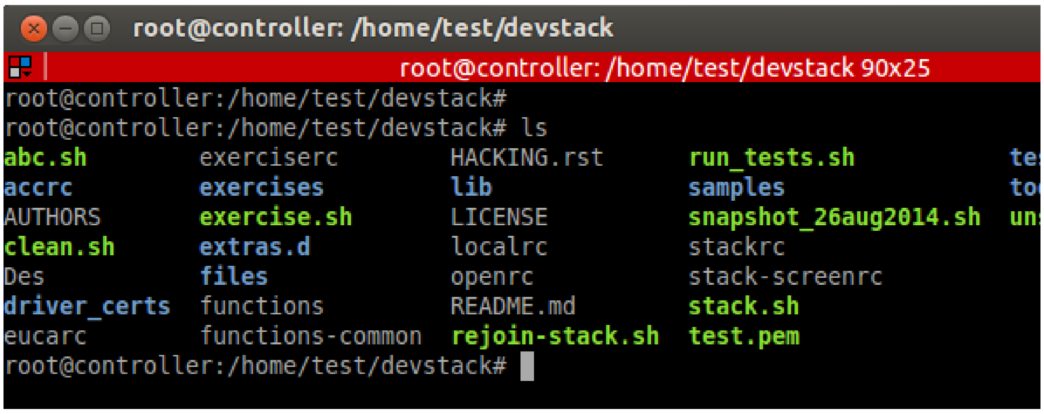
```
sudo apt-get update
sudo apt-get install git
```

To clone the DevStack repository, use the following command

```
sudo git clone https://github.com/openstack-dev/devstack.git
```

And then navigate to devstack root directory which contains stack.sh script which will be used for installation. Devstack root directory is also shown in Figure A.1.

```
cd devstack
```



```
root@controller: /home/test/devstack
root@controller: /home/test/devstack 90x25
root@controller:/home/test/devstack#
root@controller:/home/test/devstack# ls
abc.sh        exerciserc        HACKING.rst     run_tests.sh              te
accrc         exercises         lib             samples                   to
AUTHORS       exercise.sh       LICENSE         snapshot_26aug2014.sh     un
clean.sh      extras.d          localrc         stackrc
Des           files             openrc          stack-screenrc
driver_certs  functions         README.md       stack.sh
eucarc        functions-common  rejoin-stack.sh test.pem
root@controller:/home/test/devstack#
```

Figure A.1: DevStack local repository on VM

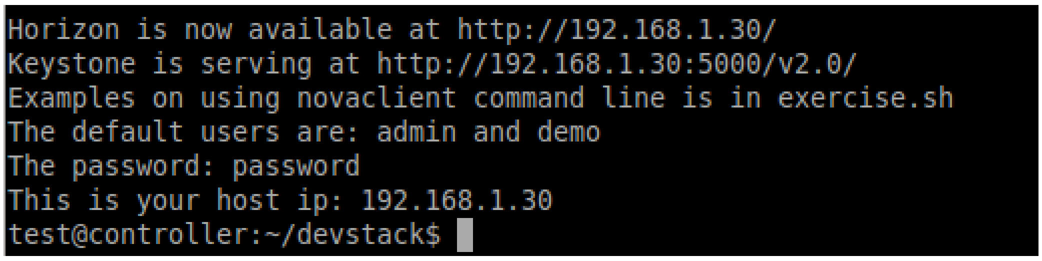
We need to change some setting in devstack configuration file before stack.sh execution. We have to modify localrc with following given setting.we are using almost all services of OpenStack for SV2M. The localrc file is located in under devstack which contains some of the configuration parameters and can be modified as-per-need user need. If localrc does not exist in devstack main directory then create it otherwise change the existing which must contain following lines.

```
ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=tokentoken
#SCHEDULER=nova.scheduler.simple.SimpleScheduler
LOGFILE=/opt/stack/data/stack.log
SCREEN_LOGDIR=/opt/stack/data/log
#RECLONE=yes
disable_service swift
HOST_IP=192.168.1.30
FIXED_RANGE=10.4.128.0/20
FIXED_NETWORK_SIZE=4096
FLOATING_RANGE=192.168.100.0/24
MULTI_HOST=1
```

Now go to devstack main directory, execture stack.sh scripts using the following command and wait for few minute for easy installation of OpenStack.

```
./stack.sh
```

After the successful execution of script, Following output Figure A.2 will be shown to you.



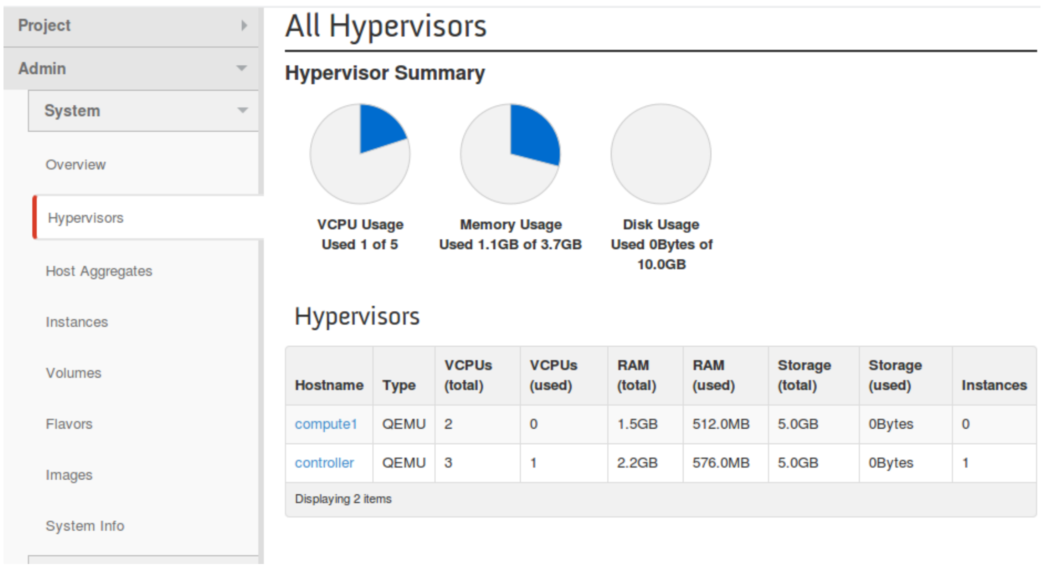
```
Horizon is now available at http://192.168.1.30/
Keystone is serving at http://192.168.1.30:5000/v2.0/
Examples on using novaclient command line is in exercise.sh
The default users are: admin and demo
The password: password
This is your host ip: 192.168.1.30
test@controller:~/devstack$
```

Figure A.2: Devstack based OpenStack Cloud

After this, you have a deployment of OpenStack on VM and to run new instance on it. OpenStack can be accessed from CLI and GUI. Please type https;//192.168.1.30/ in the browser to access the GUI for the further configuration of Opestack. GUI of installed OpenStack Cloud is shown in Figure A.3

A.1.1 Setup environment for SV2M

Now we have OpenStack Cloud running on VM. To run SV2M with Openstack we required more packages/software on this VM. There software are



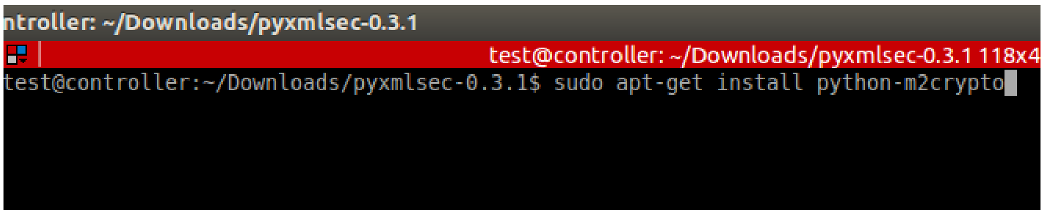
Hostname	Type	VCPUs (total)	VCPUs (used)	RAM (total)	RAM (used)	Storage (total)	Storage (used)	Instances
compute1	QEMU	2	0	1.5GB	512.0MB	5.0GB	0Bytes	0
controller	QEMU	3	1	2.2GB	576.0MB	5.0GB	0Bytes	1

Displaying 2 items

Figure A.3: OpenStack GUI

required by the different modules of SV2M to work properly. First of all, one of SV2M modulle called as Certificate management module (CMM) used for generation of certificates for authentication module. We are using M2Crypto library for RSA key generation of 2048 bit, certificate generation in X509 format and finally signing these certificate from trusted third party (TTP) for the use in mutual authentication module. Type following given command in terminal for the installation of M2Crypto package in Ubuntu. It is shown in Figure A.4

```
sudo apt-get install python-m2crypto
```



```
ntroller: ~/Downloads/pyxmlsec-0.3.1
                              test@controller: ~/Downloads/pyxmlsec-0.3.1 118x4
test@controller:~/Downloads/pyxmlsec-0.3.1$ sudo apt-get install python-m2crypto
```

Figure A.4: M2Crypto installation in Ubuntu

Mutual Authentication Module (MAM) of SV2M is used for providing mutual agreement between source and destination Cloud for Secure VM migration. It uses the same certificate generated by CMM module for source and destination Cloud. In this module we implemented FIPS-196 standard which provides this security features.

The FIPS-196 mutual authentication protocol refers involved entities Cloud A and Cloud B as "initiator" and "responder". Each entity of this protocol acts as both a claimant and a verifier in the protocol which is shown in Figure A.5

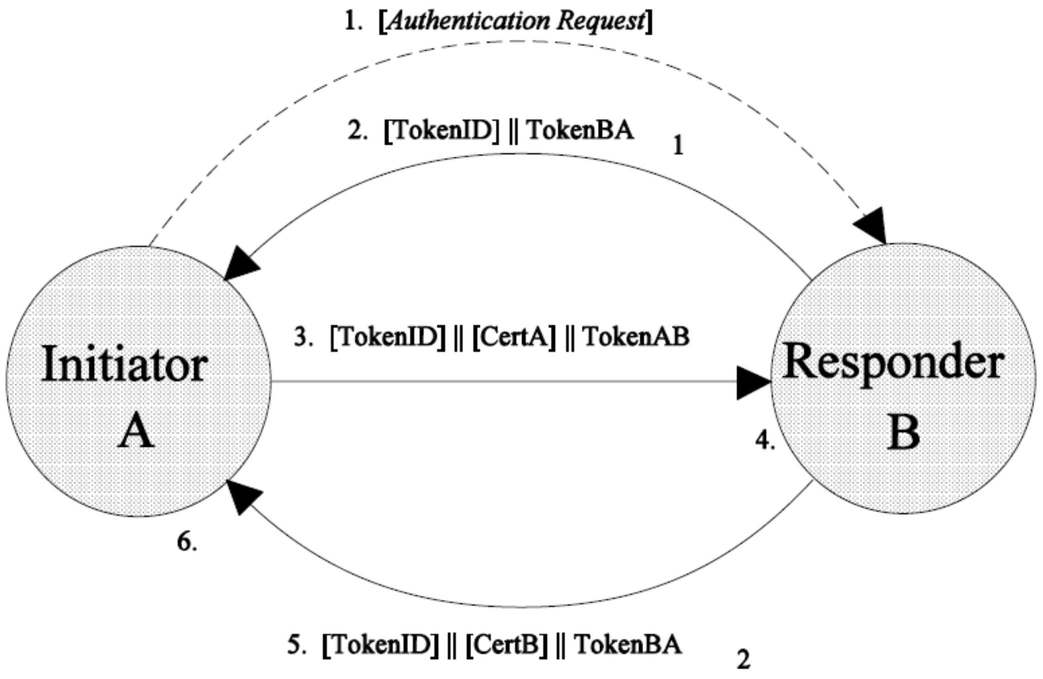

1. [*Authentication Request*]
2. [TokenID] || TokenBA
1
3. [TokenID] || [CertA] || TokenAB
Initiator A
Responder B
4.
6.
5. [TokenID] || [CertB] || TokenBA
2

Figure A.5: FIPS-196 Mutual Authentication Protocol

The authentication of an Cloud entity include following two things:

1. The verification of the claimant's ID binding with its key pair
2. The verification of the claimant's generated digital signature on the random number challenge.

For this work, destination Cloud act as the server and sender side act as Client of FIPS196 implementation. We will discuss more about the implementation and usage of scripts in next section.

As we already discuss that we have used XML encryption/digital signature for security feature of SV2M. XML security features in python are provided by PyXMLSec library. It is a set of Python bindings for the XML Security Library. XML Security Library is implemented in C programming language and it is based on LibXML2. The current version of library supports following XML security standards:

1. XML Signature
2. XML Encryption
3. Canonical XML
4. Exclusive Canonical XML

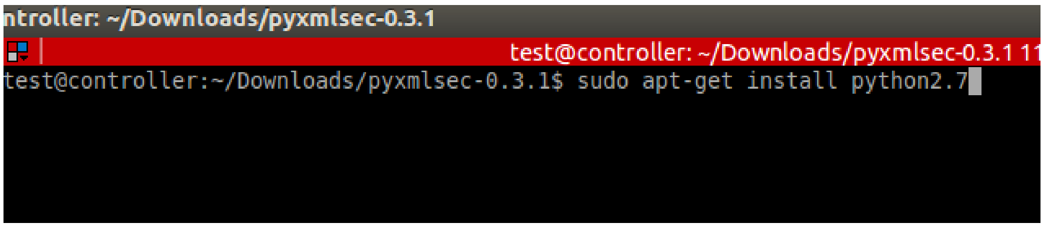
PyXMLSec tool is dependent on following software so first we have to install these packages.

Requirements

1. Python 2.2 or greater installed
2. LibXML
3. XML Security Library

Usually Python is pre installed on Ubuntu distribution of Linux. However if it is not already installed then use following commands for the installation of python 2.7 version. It is shown in Figure A.6

```
sudo apt-get install python2.7
```



```
ntroller: ~/Downloads/pyxmlsec-0.3.1
test@controller: ~/Downloads/pyxmlsec-0.3.1 11
test@controller:~/Downloads/pyxmlsec-0.3.1$ sudo apt-get install python2.7
```

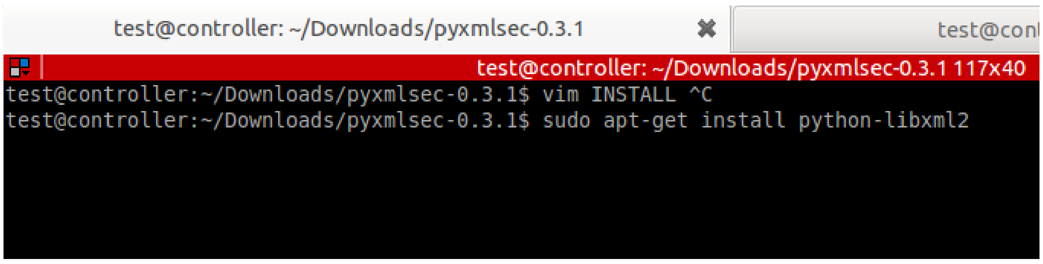
Figure A.6: Python installation

LibXML package can be installed by typing following command in the terminal. LibXML installation is shown Figure A.7

```
sudo apt-get install python-libxml2
```

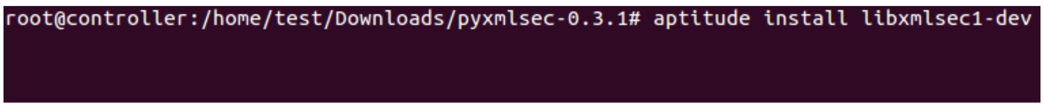
XML security library is the final requirement of PyXMLSec package. It can be installed by using following command in the terminal. Installation of XML library is shown Figure A.8 ,Figure A.9 & Figure A.10

```
sudo apt-get install libxmlsec1-dev
sudo apt-get install libxml-security-c-dev
sudo apt-get install xmlsec1
```


test@controller: ~/Downloads/pyxmlsec-0.3.1
test@con
test@controller: ~/Downloads/pyxmlsec-0.3.1 117x40

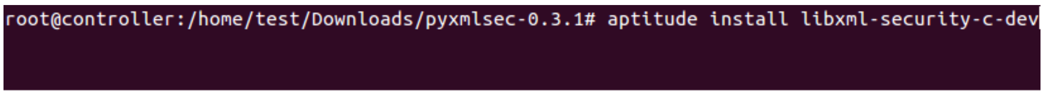
```
test@controller:~/Downloads/pyxmlsec-0.3.1$ vim INSTALL ^C
test@controller:~/Downloads/pyxmlsec-0.3.1$ sudo apt-get install python-libxml2
```

Figure A.7: LibXML Library for PyXMLSec



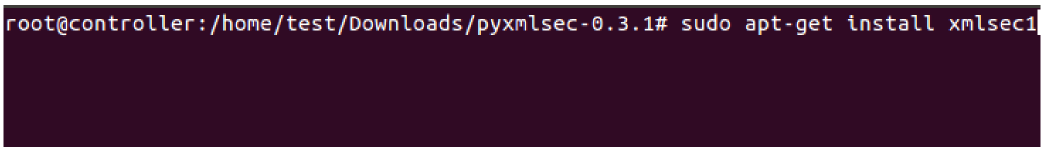
```
root@controller:/home/test/Downloads/pyxmlsec-0.3.1# aptitude install libxmlsec1-dev
```

Figure A.8: Development package for XML Security Library in Python



```
root@controller:/home/test/Downloads/pyxmlsec-0.3.1# aptitude install libxml-security-c-dev
```

Figure A.9: C Library for XML Security

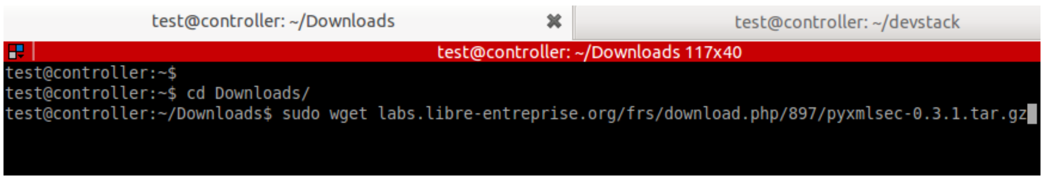


```
root@controller:/home/test/Downloads/pyxmlsec-0.3.1# sudo apt-get install xmlsec1
```

Figure A.10: XML Security Library in Python

Once successful installation of all above dependencies, now download PyXMLsec recent version (0.3.1) using following command in Downloads directory.Deb package of PyXMLsec is not available, so we have to compile it from source. Figure A.11 showing the downloading process.

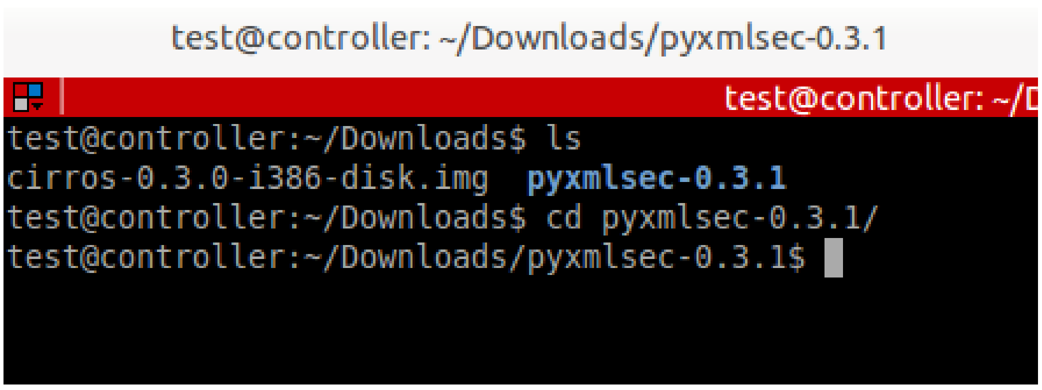
```
cd Downloads
sudo wget labs.libre-entreprise.org/frs/download.php/897/pyxmlsec-0.3
```


test@controller: ~/Downloads
test@controller: ~/devstack
test@controller: ~/Downloads 117x40

```
test@controller:~$
test@controller:~$ cd Downloads/
test@controller:~/Downloads$ sudo wget labs.libre-entreprise.org/frs/download.php/897/pyxmlsec-0.3.1.tar.gz
```

Figure A.11: Downloading PyXMLSec

After successful download, extract tar.gz file and then go to newly created directory pyxmlsec-0.3.1 which is shown in the below Figure A.12

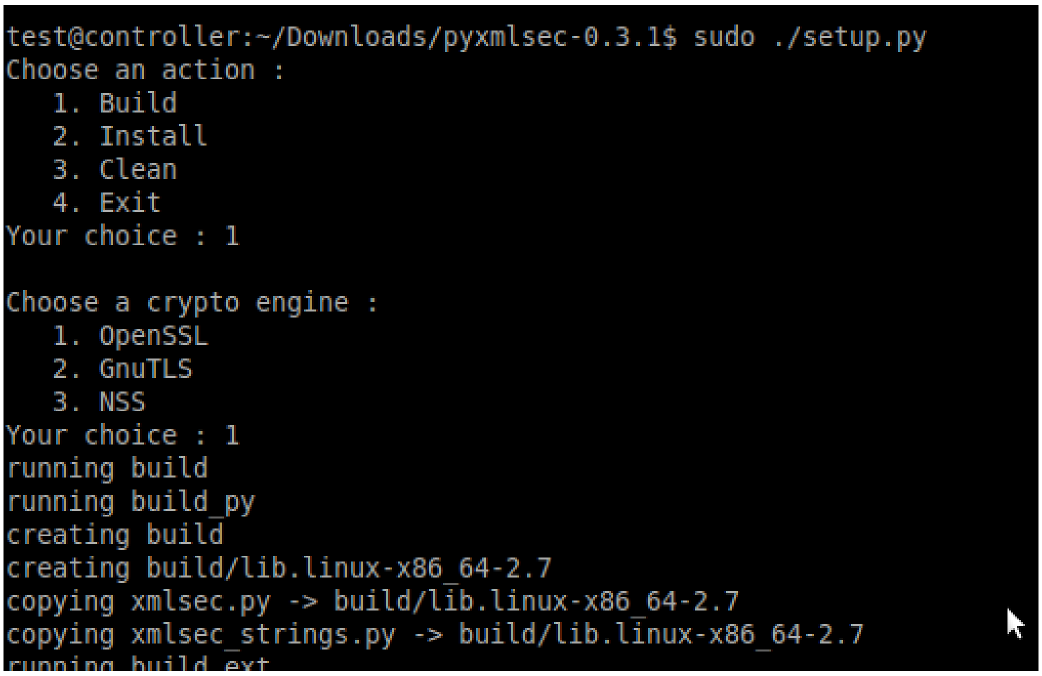


```
test@controller: ~/Downloads/pyxmlsec-0.3.1
test@controller:~/Downloads$ ls
cirros-0.3.0-i386-disk.img  pyxmlsec-0.3.1
test@controller:~/Downloads$ cd pyxmlsec-0.3.1/
test@controller:~/Downloads/pyxmlsec-0.3.1$
```

Figure A.12: PyXMLSec 0.13.1

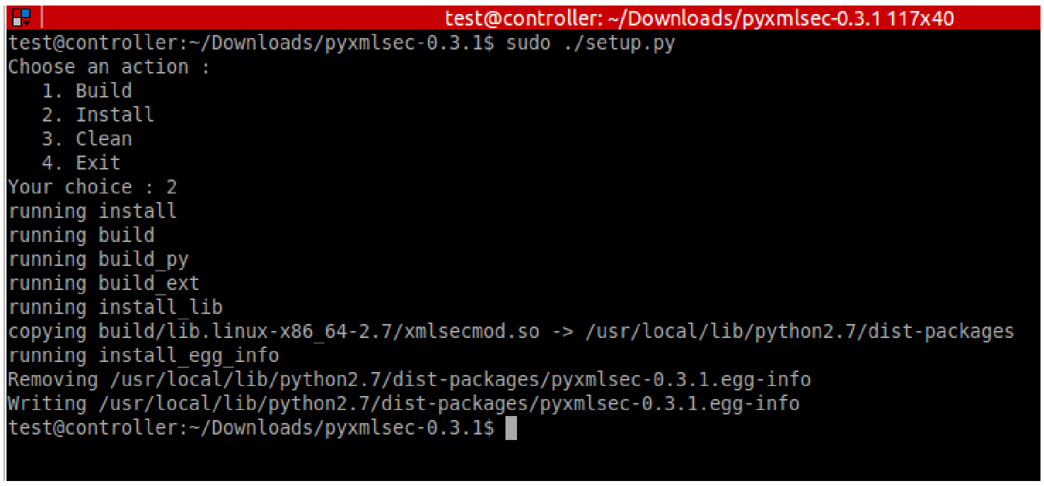
Now type setup.py in terminal to install PyxmlSec. PyXMLsec installation process is shown in Figure A.13 & Figure A.14

```
cd pyxmlsec0.13.1
sudo ./setup.py
```



```
test@controller:~/Downloads/pyxmlsec-0.3.1$ sudo ./setup.py
Choose an action :
   1. Build
   2. Install
   3. Clean
   4. Exit
Your choice : 1

Choose a crypto engine :
   1. OpenSSL
   2. GnuTLS
   3. NSS
Your choice : 1
running build
running build_py
creating build
creating build/lib.linux-x86_64-2.7
copying xmlsec.py -> build/lib.linux-x86_64-2.7
copying xmlsec_strings.py -> build/lib.linux-x86_64-2.7
running build_ext
```

Figure A.13: PyXMLSec 0.13.1 Installation 1



```
test@controller: ~/Downloads/pyxmlsec-0.3.1 117x40
test@controller:~/Downloads/pyxmlsec-0.3.1$ sudo ./setup.py
Choose an action :
   1. Build
   2. Install
   3. Clean
   4. Exit
Your choice : 2
running install
running build
running build_py
running build_ext
running install_lib
copying build/lib.linux-x86_64-2.7/xmlsecmod.so -> /usr/local/lib/python2.7/dist-packages
running install_egg_info
Removing /usr/local/lib/python2.7/dist-packages/pyxmlsec-0.3.1.egg-info
Writing /usr/local/lib/python2.7/dist-packages/pyxmlsec-0.3.1.egg-info
test@controller:~/Downloads/pyxmlsec-0.3.1$
```

Figure A.14: PyXMLSec 0.13.1 Installation 2

A.1.2 Running SV2M scripts

This section is about the running of SV2M different scripts. Extract SV2M.tar.gz on the Destkop and it consist of two directory, fips196 is for certificate generation and mutual authentication and encrypt directory of SV2M is for the remaining security features.

A.1.2.1 Certificate Generation and Mutual Authentication

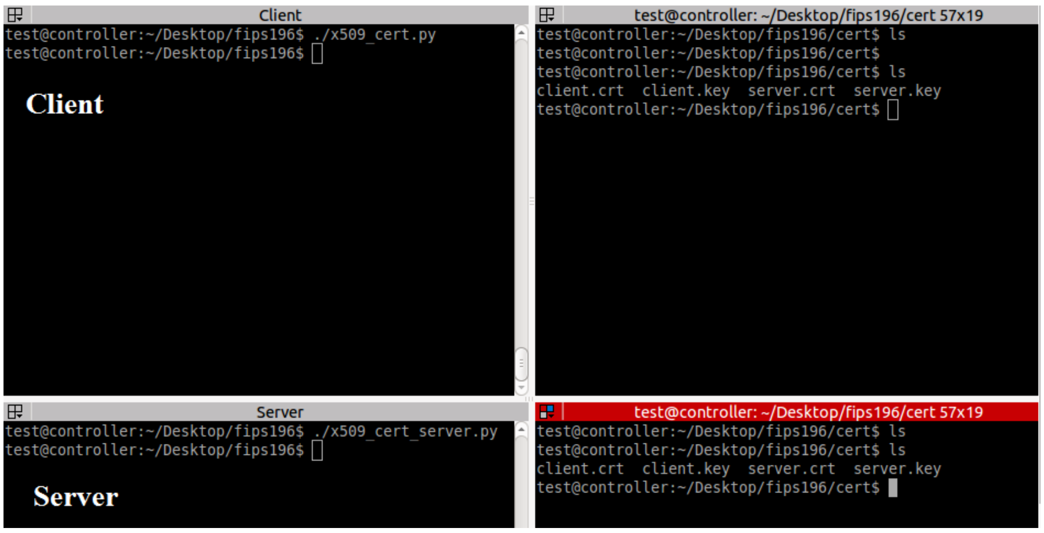
First of all, change the permission of fips196 directory using following command

```
cd /home/test/Desktop
sudo chmod +x fips196 -R
```

Now go inside the fips196 directory. X509_cert.py scripts generates the certificate for Source Cloud (Client) and X509_cert_server.py generates the certificate for Destination Cloud (Server). Certificate generation for client/server is shown in Figure A.15

These certificate are now available for Mutual Authentication Module of SV2M. In this module server.py (receiver CSP) listens form the incoming request from client.py (sender CSP). Both, CSP send thir random numbers, certificates and finally validate each other. Working of mutual authentication is shown in Figure A.16 & Figure A.17

After successful execution of mutual authentication, next step is creation of new VM and then migrate it securely. Now using GUI of Opnestack, we have to create VM with name **ab** and flavor **m1.nano** which is used


Client
test@controller: ~/Desktop/fips196/cert 57x19
Client
Server
test@controller: ~/Desktop/fips196/cert 57x19
Server

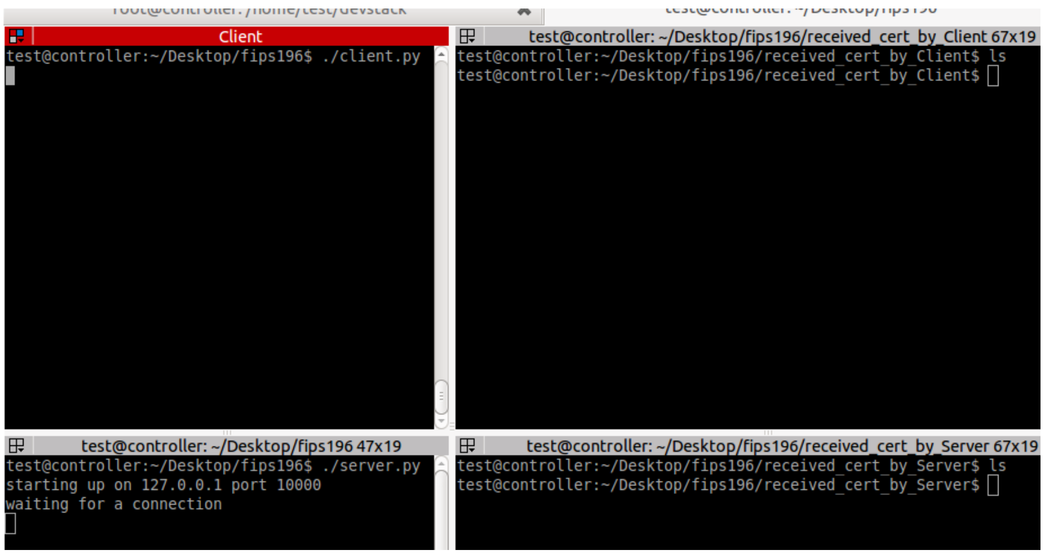
```
test@controller:~/Desktop/fips196$ ./x509_cert.py
test@controller:~/Desktop/fips196$
```

```
test@controller:~/Desktop/fips196/cert$ ls
test@controller:~/Desktop/fips196/cert$
test@controller:~/Desktop/fips196/cert$ ls
client.crt  client.key  server.crt  server.key
test@controller:~/Desktop/fips196/cert$
```

```
test@controller:~/Desktop/fips196$ ./x509_cert_server.py
test@controller:~/Desktop/fips196$
```

```
test@controller:~/Desktop/fips196/cert$ ls
test@controller:~/Desktop/fips196/cert$ ls
client.crt  client.key  server.crt  server.key
test@controller:~/Desktop/fips196/cert$
```

Figure A.15: Certificate Generation in SV2M


Client
test@controller: ~/Desktop/fips196/received_cert_by_Client 67x19
test@controller: ~/Desktop/fips196 47x19
test@controller: ~/Desktop/fips196/received_cert_by_Server 67x19

```
test@controller:~/Desktop/fips196$ ./client.py
```

```
test@controller:~/Desktop/fips196/received_cert_by_Client$ ls
test@controller:~/Desktop/fips196/received_cert_by_Client$
```

```
test@controller:~/Desktop/fips196$ ./server.py
starting up on 127.0.0.1 port 10000
waiting for a connection
```

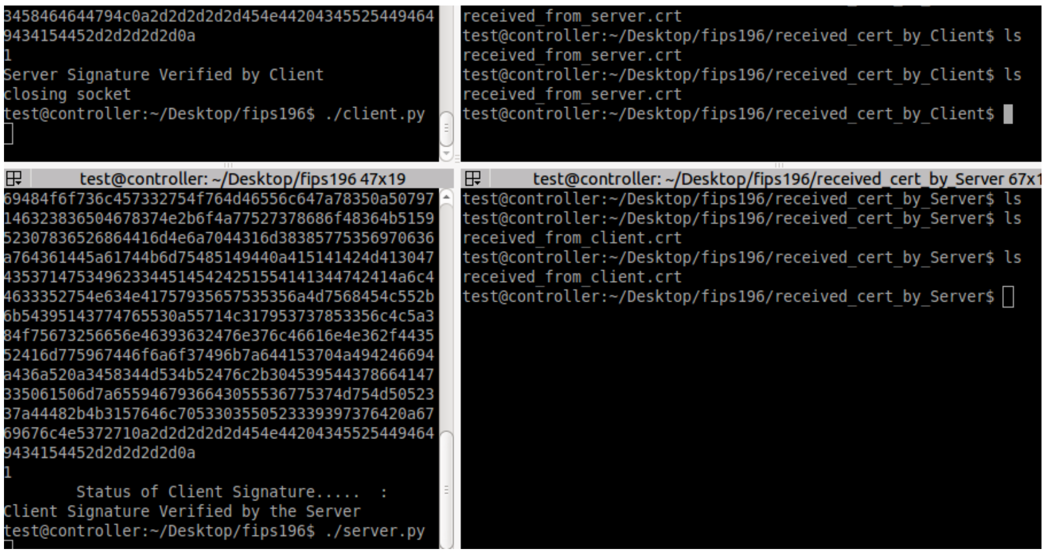
```
test@controller:~/Desktop/fips196/received_cert_by_Server$ ls
test@controller:~/Desktop/fips196/received_cert_by_Server$
```

Figure A.16: Server Cloud is listening in SV2M

by secure VM migration bash script. Creation of new instance is shown in Figure A.18 & Figure A.19.

OpenStack can be managed using commands as well. So we can see the status of new instance using CLI by typing following command in the terminal which is also shown in Figure A.20

```
cd devstack
source openrc admin admin
nova list
```



```
3458464644794c0a2d2d2d2d2d454e44204345525449464
9434154452d2d2d2d2d0a
1
Server Signature Verified by Client
closing socket
test@controller:~/Desktop/fips196$ ./client.py
```

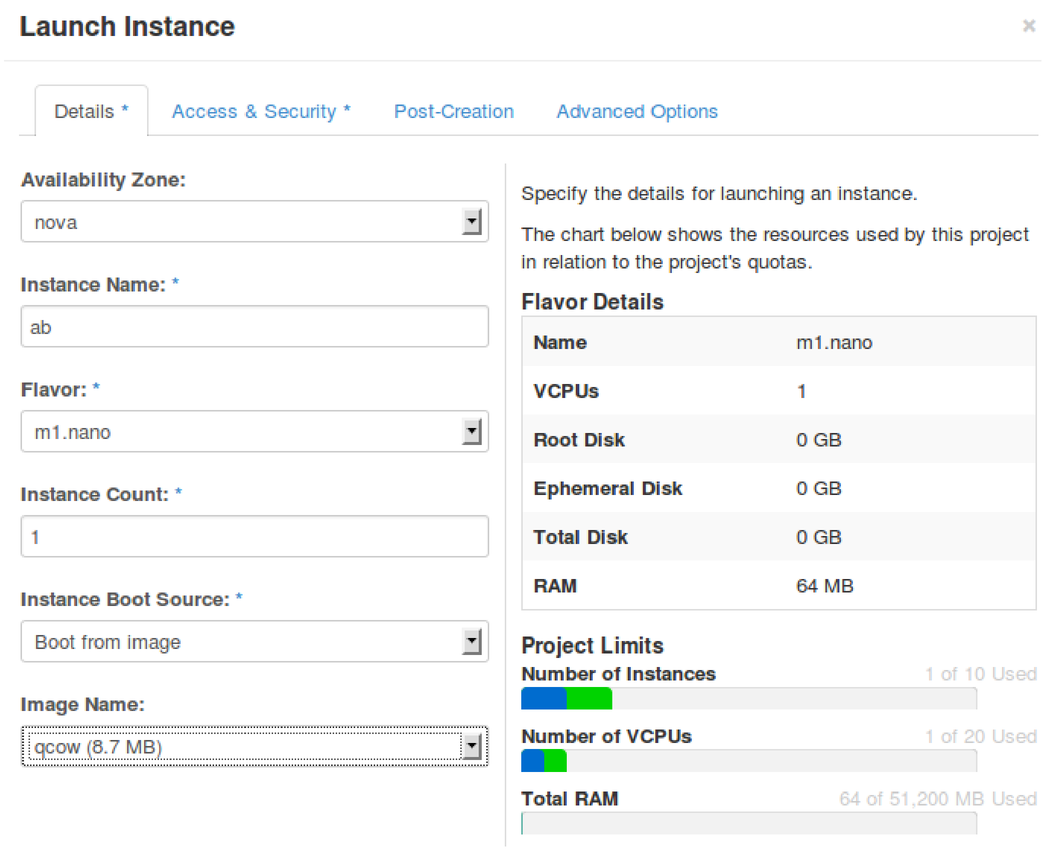
```
received_from_server.crt
test@controller:~/Desktop/fips196/received_cert_by_Client$ ls
received_from_server.crt
test@controller:~/Desktop/fips196/received_cert_by_Client$ ls
received_from_server.crt
test@controller:~/Desktop/fips196/received_cert_by_Client$
```

```
69484f6f736c457332754f764d46556c647a78350a50797
1463238365046783743e2b6f4a77527378686f48364b5159
5230783652686441 6d4e6a7044316d3838577535697063 6
```

test@controller: ~/Desktop/fips196/received_cert_by_Server 67x1

```
test@controller:~/Desktop/fips196/received_cert_by_Server$ ls
test@controller:~/Desktop/fips196/received_cert_by_Server$ ls
received_from_client.crt
test@controller:~/Desktop/fips196/received_cert_by_Server$ ls
received_from_client.crt
test@controller:~/Desktop/fips196/received_cert_by_Server$
```

Figure A.17: CSP Certificate Validation



Launch Instance

Availability Zone: nova

Instance Name: * ab

Flavor: * m1.nano

Instance Count: * 1

Instance Boot Source: * Boot from image

Image Name: qcow (8.7 MB)

Specify the details for launching an instance.

The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

Name	m1.nano
VCPUs	1
Root Disk	0 GB
Ephemeral Disk	0 GB
Total Disk	0 GB
RAM	64 MB

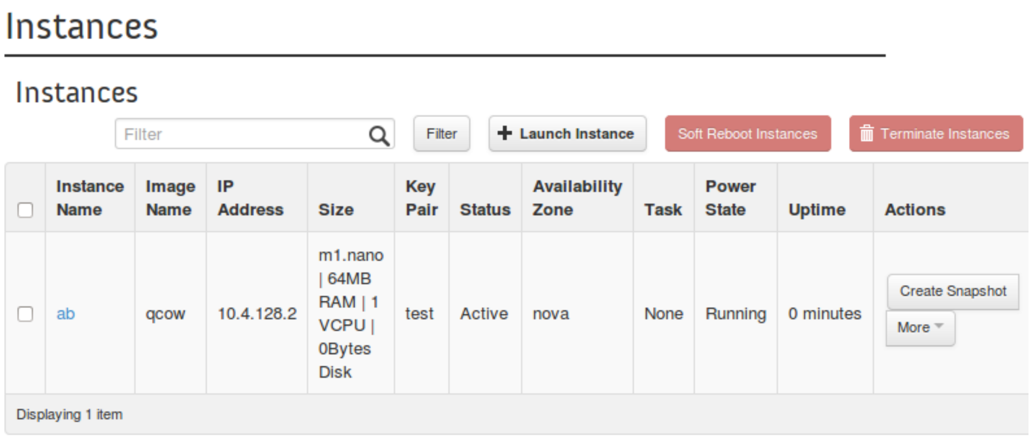
Project Limits

Number of Instances 1 of 10 Used

Number of VCPUs 1 of 20 Used

Total RAM 64 of 51,200 MB Used

Figure A.18: VM creation in OpenStack


Instances

Instances

	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Uptime	Actions
☐	ab	qcow	10.4.128.2	m1.nano | 64MB RAM | 1 VCPU | 0Bytes Disk	test	Active	nova	None	Running	0 minutes	Create Snapshot More

Displaying 1 item

Figure A.19: VM **ab** in OpenStack Dashboard

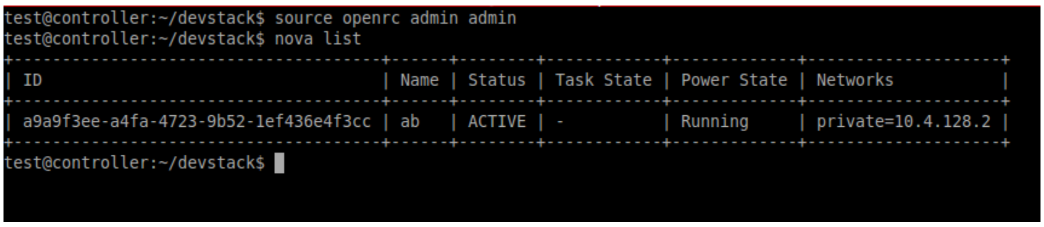

```
test@controller:~/devstack$ source openrc admin admin
test@controller:~/devstack$ nova list
+--------------------------------------+------+--------+------------+-------------+--------------------+
| ID                                   | Name | Status | Task State | Power State | Networks           |
+--------------------------------------+------+--------+------------+-------------+--------------------+
| a9a9f3ee-a4fa-4723-9b52-1ef436e4f3cc | ab   | ACTIVE | -          | Running     | private=10.4.128.2 |
+--------------------------------------+------+--------+------------+-------------+--------------------+
test@controller:~/devstack$ 
```

Figure A.20: VM **ab** status using CLI

In our thesis work, we have considered a single VM as a server and client machine for the demonstration purpose . We assume that our VM is running on client machine and due to some issue we want to migrate it to another Cloud. we have created a bash script which contains many other python scripts which provides the encryption/decryption/random key generation/digital signature functionalities. Suspension and resumption of VM are also provided by this bash script (scr_vmmigration.sh).

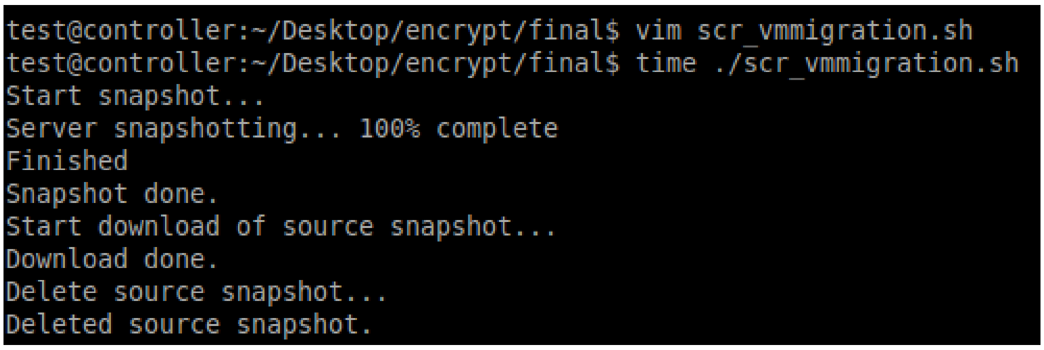
Type following command in the terminal to migrate VM **ab** which is shown in following Figure A.21.

```
cd encrypt/final
time ./scr\_vmmigration.sh
```

XML Encryption and Signature is shown in Figure A.22.

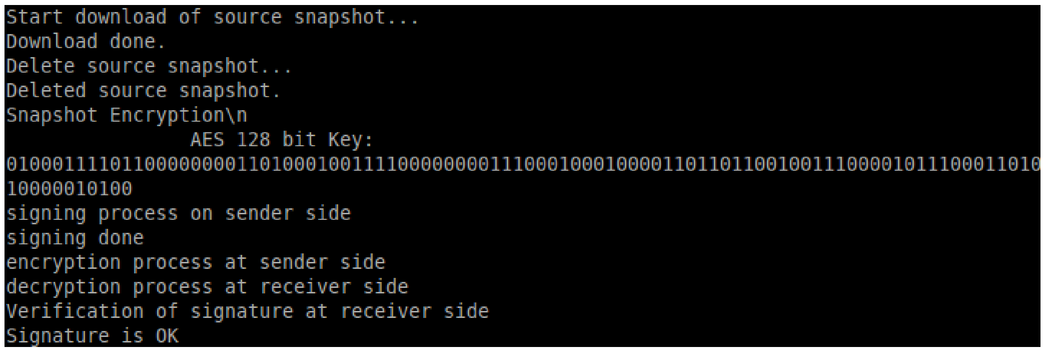
We have used time command with the script to evaluate its performance which is shown Figure A.23.

New instance with name **ab_new** is shown in Figure A.24 & Figure A.25.



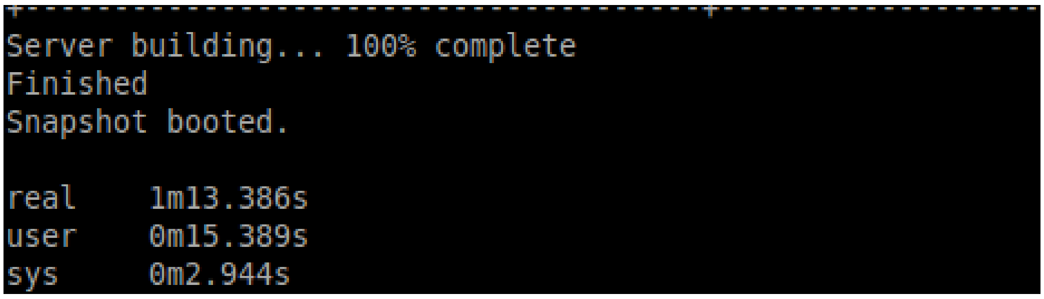
```
test@controller:~/Desktop/encrypt/final$ vim scr_vmmigration.sh
test@controller:~/Desktop/encrypt/final$ time ./scr_vmmigration.sh
Start snapshot...
Server snapshotting... 100% complete
Finished
Snapshot done.
Start download of source snapshot...
Download done.
Delete source snapshot...
Deleted source snapshot.
```

Figure A.21: Secure VM Migration script



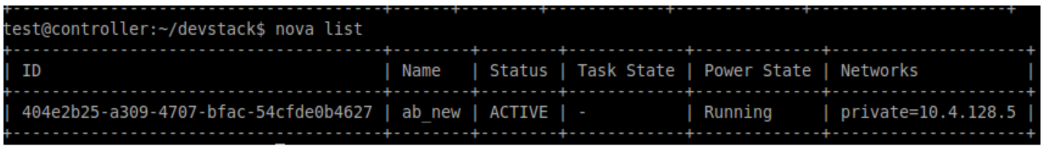
```
Start download of source snapshot...
Download done.
Delete source snapshot...
Deleted source snapshot.
Snapshot Encryption\n
                AES 128 bit Key:
0100011110110000000001101000100111100000000111000100010000110110110010011100001011100011010
10000010100
signing process on sender side
signing done
encryption process at sender side
decryption process at receiver side
Verification of signature at receiver side
Signature is OK
```

Figure A.22: Secure VM Migration Signature status



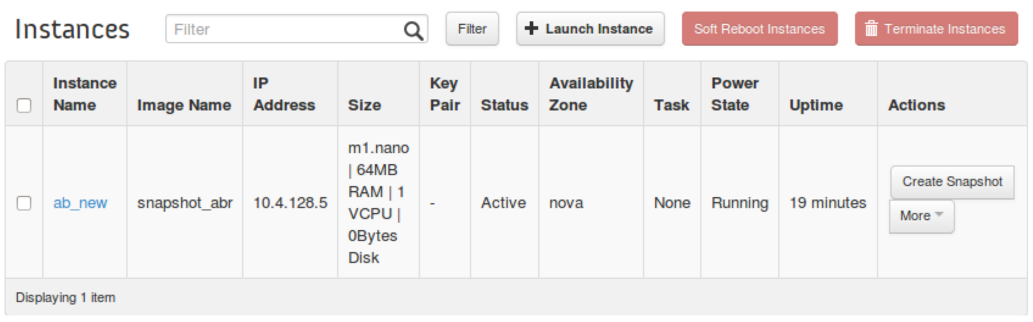
```
+------------------------------------+-----------------
Server building... 100% complete
Finished
Snapshot booted.

real    1m13.386s
user    0m15.389s
sys     0m2.944s
```

Figure A.23: Time taken by the script



```
+--------------------------------------+--------+---------+------------+-------------+---------------------+
test@controller:~/devstack$ nova list
+--------------------------------------+--------+--------+------------+-------------+--------------------+
| ID                                   | Name   | Status | Task State | Power State | Networks           |
+--------------------------------------+--------+--------+------------+-------------+--------------------+
| 404e2b25-a309-4707-bfac-54cfde0b4627 | ab_new | ACTIVE | -          | Running     | private=10.4.128.5 |
+--------------------------------------+--------+--------+------------+-------------+--------------------+
```

Figure A.24: Nova list command output



Instances Filter Filter Launch Instance Soft Reboot Instances Terminate Instances

	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Uptime	Actions
☐	ab_new	snapshot_abr	10.4.128.5	m1.nano | 64MB RAM | 1 VCPU | 0Bytes Disk	-	Active	nova	None	Running	19 minutes	Create Snapshot More

Displaying 1 item

Figure A.25: OpenStack Dashboard

A.2 Troubleshooting for Devstack

A.2.1 Devstack based OpenStack Cloud behind proxy server

Devstack behind the proxy server required following lines need to be included in the localrc file located /home/Usernmae/devstack folder. If the proxy address is 10.1.11.11 with port 8080 then lines in the locarc will be.

```
http_proxy = http://10.1.11.11:8080/
https_proxy = https://10.1.11.11:8080/
export no_proxy = "localhost,127.0.0.1"
HOST_IP=localhost
SERVICE_HOST=$HOST_IP
IMAGE_HOST=$HOST_IP
IDENTITY_HOST=$HOST_IP
```

A.2.2 Logging in Devstack

If you want to see the logs for Devstack installation then following line needs to be written in the localrc file.

```
LOGFILE=$DEST/logs/stack.sh.log
```

A.2.3 Unable to Install due to error ”11 Resource temporarily unavailable”

A common error occur on Ubuntu while installing new package from terminal. Error description is **Could not get lock /var/lib/dpkg/lock - open (11 Resource temporarily unavailable)** . To fix this issue, run the following commands in terminal.

```
sudo rm /var/lib/apt/lists/lock
sudo rm /var/cache/apt/archives/lock
```

A.2.4 Invalid Nova Credentials (Unauthorized HTTP 401) while running command in terminal

It is required to export admin user name and password in terminal to run commands for OpenStack Cloud. As we already mentioned this command in above section, first run following command in terminal before running other command for OpenStack operation.

```
cd devstack
source openrc admin admin
```

Bibliography

[1] M. Rosenblum, “The reincarnation of virtual machines,” *ACM Queue*, vol. 2, no. 5, pp. 34–40, 2004. [Online]. Available: http://doi.acm.org/10.1145/1016998.1017000

[2] S. T. King, G. W. Dunlap, and P. M. Chen, “Operating system support for virtual machines.”

[3] J. Szefer and R. Lee, “A case for hardware protection of guest vms from compromised hypervisors in cloud computing,” in *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*. IEEE, 2011, pp. 248–252.

[4] E. Ray and E. Schultz, “Virtualization security,” in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*. ACM, 2009, p. 42.

[5] P. Hoffman, K. Scarfone, and M. Souppaya, “Guide to security for full virtualization technologies,” *National Institute of Standards and Technology (NIST)*, pp. 800–125, 2011.

[6] “Crisis malware targets virtual machines,” http://www.zdnet.com/crisis-malware-targets-virtual-machines-7000002986, last Accessed: 2012-12-24.

[7] R. Shea and J. Liu, “Understanding the impact of denial of service attacks on virtual machines,” in *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*. IEEE Press, 2012, p. 27.

[8] K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernandez, “An analysis of security issues for cloud computing,” *Journal of Internet Services and Applications*, vol. 4, no. 1, pp. 1–13, 2013.

[9] P. Mell and T. Grance, “The nist definition of cloud computing (draft),” *NIST special publication*, vol. 800, no. 145, p. 7, 2011.

[10] ——, “The nist definition of cloud computing,” 2011.

[11] V. Vaidya, “Virtualization vulnerabilities and threats: A solution white paper,” *RedCannon Security, Inc*, pp. 1–7, 2009.

[12] J. Shetty, A. MR *et al.*, “A survey on techniques of secure live migration of virtual machine,” *International Journal of Computer Applications*, vol. 39, no. 12, pp. 34–39, 2012.

[13] J. Oberheide, E. Cooke, and F. Jahanian, “Empirical exploitation of live virtual machine migration,” in *Proc. of BlackHat DC convention.* Citeseer, 2008.

[14] N. Ahmad, A. Kanwal, and M. Shibli, “Survey on secure live virtual machine (vm) migration in cloud,” in *Information Assurance (NCIA), 2013 2nd National Conference on*, Dec 2013, pp. 101–106.

[15] M. A. Shibli, N. Ahmad, A. Kanwal, and A. G. Abbasi, “Secure virtual machine migration (SV2M) in cloud federation,” in *SECRYPT 2014 - Proceedings of the 11th International Conference on Security and Cryptography, Vienna, Austria, 28-30 August, 2014*, 2014, pp. 344–349. [Online]. Available: http://dx.doi.org/10.5220/0005057103440349

[16] K. Nagin, D. Hadas, Z. Dubitzky, A. Glikson, I. Loy, B. Rochwerger, and L. Schour, “Inter-cloud mobility of virtual machines,” in *Proceedings of the 4th Annual International Conference on Systems and Storage.* ACM, 2011, p. 3.

[17] Migrating virtual machines.

[18] O. Community. (2013) Pausing and suspending instances. [Online]. Available: http://docs.openstack.org/grizzly/openstack-compute/admin/content/pausing-and-suspending-instances.html

[19] X. Community. (2012) Storage xenmotion. [Online]. Available: http://wiki.xen.org/wiki/Migration

[20] ——. (2011) Xen migration.

[21] R. KVM. Kvm migration. [Online]. Available: http://www.linux-kvm.org/page/Migration

[22] Virtualization comparison matrix.

[23] X. Wan, X. Zhang, L. Chen, and J. Zhu, "An improved vtpm migration protocol based trusted channel," in *Systems and Informatics (ICSAI), 2012 International Conference on.* IEEE, 2012, pp. 870–875.

[24] O. Community. (2013) Openstack security guide. [Online]. Available: http://docs.openstack.org/security-guide/securityguide.pdf

[25] W. Wang, Y. Zhang, B. Lin, X. Wu, and K. Miao, "Secured and reliable vm migration in personal cloud," in *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, vol. 1. IEEE, 2010, pp. V1–705.

[26] B. Danev, R. J. Masti, G. O. Karame, and S. Capkun, "Enabling secure vm-vtpm migration in private clouds," in *Proceedings of the 27th Annual Computer Security Applications Conference.* ACM, 2011, pp. 187–196.

[27] Y. Chen, Q. Shen, P. Sun, Y. Li, Z. Chen, and S. Qing, "Reliable migration module in trusted cloud based on security level-design and implementation," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International.* IEEE, 2012, pp. 2230–2236.

[28] V. P. Patil and G. Patil, "Migrating process and virtual machine in the cloud: Load balancing and security perspectives," *International Journal of Advanced Computer Science and Information Technology*, vol. 1, no. 1, pp. pp–11, 2012.

[29] M. Aslam, C. Gehrmann, and M. Bjorkman, "Security and trust preserving vm migrations in public clouds," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on.* IEEE, 2012, pp. 869–876.

[30] F. Zhang, Y. Huang, H. Wang, H. Chen, and B. Zang, "Palm: security preserving vm live migration for systems with vmm-enforced protection," in *Trusted Infrastructure Technologies Conference, 2008. APTC'08. Third Asia-Pacific.* IEEE, 2008, pp. 9–18.

[31] H. Mouratidis and P. Giorgini, "Secure tropos: a security-oriented extension of the tropos methodology," *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, no. 02, pp. 285–309, 2007.

[32] C. Kothari, *Research methodology: methods and techniques.* New Age International, 2004.

[33] Anamika, “Research methodology:an introduction,” 2010, [Online; accessed April-2013].

[34] W. C. Booth, G. G. Colomb, and J. M. Williams, *The Craft of Research*, 3rd ed., 2008, ch. Thinking in Print: The Uses of Research, Public and Private.

[35] E. Heit and C. M. Rotello, “Relations between inductive reasoning and deductive reasoning,” *Journal of Experimental Psychology:Learning, Memory, and Cognition*, vol. 36, no. 3, pp. 805–812, 2009.

[36] A. Srivastava and S. B. Thomson, “Framework analysis: A qualitative methodology for applied policy research,” *JOAAG*, vol. 4, no. 2, pp. 72–79, 2009.

[37] W. Jansen, “Directions in security metrics research,” April 2009, [Online; accessed July-2013].

[38] Openstack achitecture and components. [Online]. Available: http://vivekraghuwanshi.wordpress.com/openstack-and-its-component/

[39] “M2crypto 0.22.3 : Python package index,” https://pypi.python.org/pypi/M2Crypto, last Accessed: 2014-01-16.

[40] E. A. U. P. K. Cryptography, “Federal information processing standards publication (fips pub) 196,” *US Department of Commerce, National Institute of Standards and Technology*, 1997.

[41] (2013) Openstack key manager. [Online]. Available: https://wiki.openstack.org/wiki/KeyManager

[42] M. Kazim, R. Masood, and M. A. Shibli, “Securing the virtual machine images in cloud computing,” in *Proceedings of the 6th International Conference on Security of Information and Networks*. ACM, 2013, pp. 425–428.

[43] O. Gelbukh. (2012) Openstack swift object storage. [Online]. Available: http://www.mirantis.com/blog/openstack-swift-encryption-architecture/

[44] H. S. P. Ltd. (2011) Encrypt and protect virtual machine images. [Online]. Available: http://www.net-security.org/secworld.php?id=11825

[45] ——. (2013) Secure vm backups. [Online]. Available: http://www.highcloudsecurity.com/blog/secure-vm-backups-what-is-different-and-why-should-you-care

[46] (2006) Hlpsl tutorial. [Online]. Available: http://www.avispa-project.org/package/tutorial.pdf

[47] (2006) Avispa v1.1 user manual. [Online]. Available: http://www.avispa-project.org/package/user-manual.pdf

[48] Time utility of linux/unix os.