Francois D'Ugard



MISSION CRITICAL CLOUD COMPUTING

Requirements Document

# Copyright Notice

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation (THE SOFTWARE). THE SOFTWARE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.  
  
THE SOFTWARE is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. In no event shall the authors be liable for any claim, damages, or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software. See the GNU General Public License for more details.  
  
You should have received a copy of the GNU General Public License along with this software. If not, see http://www.gnu.org/licenses.

# Abstract

Virtualization is an increasingly popular approach to manage rising information technology costs and complexity in every sector of the economy. Cloud computing allows organizations of any size to provision infrastructure resources as needed and flexibly scale technology resources to meet changing demands. Cloud providers pool hardware resources and allocate them based on the requests of their users. In order to efficiently allocate these resources providers must aggregate users of different requirements and workloads onto the same physical infrastructure. However, this approach increases the likelihood that a malicious user can collocate a VM alongside a target VM in order to extract information or disrupt its functioning in some way.

We propose a solution that can deliver mission assurance to mission-critical applications in cloud computing systems. We will do so by leveraging the unique capabilities of virtualization technology that and develop a dynamic and distributed approach to run applications with good security and reliability in typical cloud computing systems.

Our approach relies on developing a complete network graph on virtual private network peer to peer connections. With the purpose of reducing the likelihood of a malicious VM locating the mission-critical VM and achieving co-residency with it.

This project will deliver mission assurance to mission-critical applications in cloud computing systems. We will do so by leveraging the unique capabilities that develop a virtual machine based approach to run applications with good security and reliability in typical cloud computing systems. This project will build upon the previous project's results namely a P2P overlay network that interconnect the OpenStack VMs based on the IP-over-P2P (IPOP) framework. The project will focus on developing an extension to IPOP that will allow for the communications among the VMs to be routed by an overlay network in an OpenStack-based cloud system.

This document provides a description the software system to be developed. It includes functional and non-functional requirements.

Table of Contents

[Copyright Notice 1](#_Toc397940240)

[Abstract 2](#_Toc397940241)

[1. Introduction 4](#_Toc397940242)

[1.1. Problem Definition 5](#_Toc397940243)

[1.2. Scope of System 6](#_Toc397940244)

[1.3 Overview of Document 7](#_Toc397940245)

[2. Current System 8](#_Toc397940246)

[3. Project Plan 9](#_Toc397940247)

[3.1. Project Organization 9](#_Toc397940248)

[3.2. Work breakdown 9](#_Toc397940249)

[3.3 Cost Estimate 11](#_Toc397940250)

[4. Proposed System Requirements 12](#_Toc397940251)

[4.1. High-level Definition of User Requirements 12](#_Toc397940252)

[4.2 Analysis of System Requirements 12](#_Toc397940253)

[4.2.1 Scenarios 12](#_Toc397940254)

[4.2.2 Use case model 12](#_Toc397940255)

[4.2.3 Static model e.g., object diagrams, class diagram 12](#_Toc397940256)

[4.2.4 Dynamic model e.g., sequence diagrams or state machines 12](#_Toc397940257)

[5. Glossary 13](#_Toc397940258)

[6. Appendix 14](#_Toc397940259)

[6.1 Appendix A - Complete use cases 14](#_Toc397940260)

[6.2 Appendix B - Use case diagram using UML 14](#_Toc397940261)

[6.3 Appendix C - Static UML diagram 14](#_Toc397940262)

[6.4 Appendix D - Dynamic UML diagrams 14](#_Toc397940263)

[6.5 Appendix E - User Interface designs. 14](#_Toc397940264)

[6.6 Appendix F - Diary of meeting and tasks. 14](#_Toc397940265)

[7. References 15](#_Toc397940266)

# Introduction

Virtualization is an increasingly popular approach to manage rising information technology costs and complexity in every sector of the economy. Cloud computing allows organizations of any size to provision infrastructure resources as needed and flexibly scale technology resources to meet changing demands. Cloud providers like Amazon, Microsoft, or Rackspace pool hardware resources such as compute, memory, and storage and allocate them based on the provisioning requests of their users and available resources. In order to efficiently allocate these resources providers must necessarily aggregate users of different requirements and workloads onto the same physical infrastructure.

The rising popularity and diffusion of hardware virtualization technology among organizations of every size has led researchers and technology professionals to seriously consider the security of such computer systems. By allocating the virtual machines of different user onto the same infrastructure various security vulnerabilities are introduced. However many of the unique capabilities of cloud computing can also be used to increase the reliability of the cloud as well as ensure its security and resilience to an attack.

An important quality of virtualization is the ability for users to virtualize the underlying communications network. This capability lends itself well to the concept of network masquerading wherein the behavior and routing path of packets is managed in order to disguise the flow of network traffic within the network. This capability can be used to ensure the security of a cloud computing environment.

This chapter presents the problem definition in section 1.1 followed by a description of the scope of the system in section 1.2. Section 1.3 presents a list of common abbreviations and acronyms that are in use throughout the document. We present the overview of this document in section 1.4.

## Problem Definition

The rising popularity of cloud computing concepts has introduced significant security vulnerabilities into the information technology infrastructure of many organizations. Currently virtualized information technology systems are designed and built to operate on relatively static configurations. Typically cloud services providers fulfill provisioning requests on an as needed basis. In order for cloud computing to remain cost effective infrastructure providers must provision pools of resources, such as CPU, memory, and storage among many different users.

This configuration means that a malicious user can reside alongside a virtual machine that computes mission critical or sensitive data. The sharing of physical infrastructure among virtual machines belonging to different users opens up the possibility of a side-channel attack. A side channel attack occurs when a malicious users is able to locate a target virtual machine and spawn another virtual machine alongside this target. The malicious user is then able to extract information by monitoring the hardware resources shared by both virtual machines.

## Scope of System

We propose a distributed peer to peer virtual private network that maintains the communication network structure of collaborating virtual machines unchanged allowing a decoupling between networking services within the cloud framework and the communications network used by the virtual machines. In order to increase the resiliency of said network we propose a system that masks the communication between the virtual machines, obfuscating the communication patterns of virtual machines. Our system will be leveraging and extending IP over P2P (IPOP), a virtual network software that allows users to create virtual private networks (VPNs). IPOP will be extended such that it will allow users to join a VPN with the VMs of their. This will entail functionality to join the VPN, leave it, and communicate information correctly between VMs.

Our system should be easy to understand, setup, and use. This will require developing a command-line tool to access the system’s features as well as implementing the service so that it integrates well among existing OpenStack modules.

## . Overview of Document

This document is organized into various different sections. Chapter 1 is an introductory section. Section 1.1 defines and describes the problem we will solve. 1.2 provides some background information related to the problem including previous research. Section 1.3 we define and explain various terms and abbreviations found in this document.

Chapter 2 is a study into the feasibility of the proposed system. Section 2.1 describes the current system in place and identifies its current limitations and constraints. Section 2.2 presents the purpose of the system in detail. Section 2.3 is an overview of the user requirements that must be met upon the successful completion of the project. Based on those requirements section 2.4 compares other possible solutions and based on this analysis section 2.5 presents our recommended solution.

Chapter 3 is the project plan. The project plan describes the organization of the project in section 3.1 Also included in section 3.1.1 is the project personnel organization and hard ware and software resources in section 3.1.2. Lastly section 3.2 clearly states that tasks, milestones and deliverables expected upon completion of the project.

Chapter 4 is the appendix. It includes a project schedule in section 4.1. The feasibility matrix in section 4.2 a cost matrix section 4.3 and a diary of meetings in section 4.4

In Chapter 5 we present a glossary of the domain specific terms to facilitate the reader’s understanding. In Chapter 6 we include all appendices referred to in the document and the bibliographic references to the cited sources in Chapter 7.

# Current System

Currently virtualized information technology systems are designed and built to operate on static configurations. Typically cloud services providers fulfill provisioning requests as requested by their user base. In order for cloud computing to remain cost effective infrastructure providers must provision pools of resources, such as CPU, memory, and storage among many different users. This policy, while cost effective and efficient, opens up the cloud to the possibility of a side channel attack. By mapping the internal structure of the cloud environment a malicious attacker is able to identify a target virtual machine a spawn a malicious virtual machine co-resident to the target. This allows the malicious virtual machine to monitor the shared physical infrastructure and extract sensitive or mission critical data.

Distributed applications require a solution that will guarantee reliable and resilient communication. The current IPOP architecture address this by creating a virtual private network over a peer to peer overlay that allows the virtual machine to communicate using static virtual IP addresses. These ip address should remain constant even when changes in the cloud network, like frequent migrations or failures, take place. However the standard IPOP controllers are not designed to masquerade network traffic among cooperative virtual machines. In particular, current controllers manage link creation and termination in a reactive fashion and do not offer solutions to randomly route packets among the cooperating virtual machines. On the contrary current IPOP controllers are designed with efficiency in mind, creating links of shortest path with no consideration to network latencies and masquerading.

# Project Plan

This chapter presents the project organization including the personnel assigned to the project, and the mentor and client. We also specify the hardware and software resources required to complete the project. Lastly we identify the tasks, milestones and deliverables of our project and include a timeline of their completion dates.

## 3.1. Project Organization

This section describes the organization of all interested parties participating in the project. This section defines the roles and work domains of all interested parties in order to facilitate effective communication among the team members and the client. Also included in this section is a description of the hardware and software resources required for the successful completion of the project.

|  |  |
| --- | --- |
| **Team member** | **Main work domain** |
| Francois D’Ugard | MC^2 Controller, Testing, Implementation, Integration. |
| Ming Zhao PhD. | Consultant Guidance |
| Renato Figueiredo PhD. | Consultant - IPOP |

Figure 3.1 - Team member work domains

|  |  |
| --- | --- |
| **Team member** | **Role** |
| Francois D’Ugard | Team lead, system requirements analyst, UML modeler, Minutes taker, Time Keeper. |
| Ming Zhao PhD. | Mentor and Client |

Figure 3.2 - Team member roles

## 3.2. Work breakdown - Identification of Milestones, and Deliverables

All tasks are identified with a unique string in the format of component number, milestone number, and task number i.e. C#M#T#.

* Setup Development Environment
  + Milestone 1: Complete setup of development environment and network configuration and setup NFS file system to support live migration capabilities of OpenStack.
    - C1M1T1: Install Hardware Components
    - C1M1T2: Install Ubuntu 12.04 on all development machines
    - C1M1T3: Configure Development Network
    - C1M1T4: Deploy OpenStack cloud to all development machines
* Communications Network: Design and implementation of a distributed P2P VPN communications network that obfuscates communications with collaborating virtual machines and prioritizes communication reestablishment by communication history.
  + Milestone 2: Design distributed collaborative communication network using IPOP P2P VPN.
    - C3M2T1: Design modifications to IPOP controller
    - C3M2T3: Design communications obfuscation algorithm to hide mission critical communications between collaborating virtual machines.
  + Milestone 3: Implement Communications Network Component
    - C3M3T1: Implement Communications Network link reestablishment Algorithm
    - C3M3T2: Implement Communications Network obfuscation Algorithm
    - C3M3T3: Implement Command line interface for Communications Network component
  + Milestone 4: Integrate Communications Network Component into OpenStack system
    - C3M4T1: Systems integration and testing.

## 3.3 Cost Estimate

|  |  |  |
| --- | --- | --- |
| **Resource** | **Quantity** | **Cost** |
| Server | 2 | $5,000 |
| Personal Computer | 1 | $1,400 |
| Software | Open Source | $0 |
| Personnel (Hours @ $40) | 480 | $19,200 |
| **TOTAL** |  | **$30,600** |

Figure 3.3 - Cost Estimate Breakdown

# Proposed System Requirements

This chapter covers the requirements of the proposed system. We start by presenting a high-level description of the system in the form of functional and nonfunctional requirements in section 4.1. In section 4.2 we delve into the details of the system requirements by developing the analysis models, which define the complete functional specification of the system. In particular, in this section we describe the diagrams in the Appendices B, C, and D and validate the models against the uses cases listed in Appendix A.

## 4.1. Functional Requirements

In this section we present a high-level description of the functionality of the system in terms of functional requirements and the associated non-functional requirements for usability, reliability, performance, and/or supportability.

## 4.1.2 MC2 Controller

* The system shall allow the user to configure a VPN to interconnect a group of collaborating virtual machines while masquerading their network traffic.
  + Usability:
    - The VPN shall support the execution of unmodified applications that use the standard TCP and UDP protocols.
    - The VPN shall not impose significant constraints in the network infrastructure required for operation. In particular, the VPN shall allow communication through firewalls and NATs.
    - Initialization of the VPN shall be accessible from a command line interface useable by a computer savvy use.
    - A short description of each option executable in the system help.
    - The system shall automatically download all required files to install and run the VPN service.
  + Performance:
    - The system shall finish the setup in less than one minute after the user issues a complete and correct command on a 100 Mbps network with Internet access.
  + Supportability:
    - This feature shall function correctly in all POSIX-compliant OS.
* The system shall allow the user to join the VPN based on the configuration file previously created.
  + **Usability:** 
    - This feature shall be accessible from a command line interface simple enough for a computer savvy user to use without training.
    - A short description of each option executable in the system help.
  + **Performance:** 
    - The system shall join the VPN in less than 30 seconds after the user invokes the command with a correct configuration file on a 100 Mbps network where the XMPP server is reachable.
  + **Supportability:** 
    - This feature shall function correctly in all POSIX-compliant OS.
* The system shall allow the user to leave the VPN at any time.
  + **Usability:** 
    - This feature shall be accessible from a command line interface simple enough for a computer savvy user to use without training.
  + **Performance:** 
    - The system shall disconnect from the VPN and remove the network interface in less than ten seconds after the user invokes the command.
  + **Supportability:** 
    - This feature shall function correctly in all POSIX-compliant OS.
* The system shall allow the user to set up an open-source XMPP server for the collaborating VMs to discover each other and join the VPN.
  + **Usability:**
    - This feature shall be accessible from a command line interface simple enough for a computer savvy user to use without training.
  + **Performance:** 
    - **T**he system shall finish the setup in less than one minute after the user issues a complete and correct command on a 100 Mbps network with Internet access.
  + **Supportability:** 
    - This feature shall function correctly in all POSIX-compliant OS.

## 4.2. Analysis of System Requirements

This section contains the complete functional specification that will guide designers and programmers during the realization of the system. In particular, this section presents the elicited scenarios and describes the diagrams in the Appendices B, C, and D.

### 4.2.1. Scenarios

The following are the scenarios elicited during our interviews with the client as part of the requirements elicitation phase.

### VPN over P2P

Allow VM to communicate in dynamic cloud environment.

John wishes to deploy a Hadoop cluster across two public clouds with 100 VMs he has created in order to run a distributed hurricane simulation software.

For simplicity, John named his VM “vm-1”, “vm-2”, and so on, up to 100.

Since he will be dealing with sensitive insurance data, he decides to activate the MC2 service to improve the cluster’s resiliency. In order to create this VPN, John creates an additional VM and calls it “xmpp” and installs an XMPP server on it. Then John configures his 100 VMs to use the XMPP server. When John receives confirmation that all VMs are ready to connect to the network, he joins all VMs to the network and receives confirmation from all of them.

Disconnect VM from VPN. John’s VMs are working as expected, but he wants to make sure they can reach each other over the VPN. He logs in to vm-1 and pings the virtual IP of vm-10 to vm-15. Then John decides to remove vm-10 from the VPN and ping it again. When John realizes the VM is unreachable, he adds the VM back to the VPN.

### 4.2.2. Use Case Model

The functional model of the proposed system is captured by the use case diagram found in Appendix B. Created with the standard notation of the Unified Modeling Language (UML), this diagram describes the functionality of the proposed system from the user’s prespective.

The system we are proposing will allow cloud administrators to have access to the list of instances for which the service is enabled. The uses case that provide this functionality are start , stop, restart, join and leave.

In order to maintain the communication among the migrated VM and its collaborating peers outside the cloud, the administrator will be able to set up an XMPP server, configure the VMs to use the server. In addition, the administrator will also be able to remove nodes from the virtual network.

### 4.2.3. Static Model

The static model of the system is captured by the class diagrams in Appendix C. The vpn communication system makes use of the IPOP and MCVPN command line Interface to be able to prepare the XMPP Server that will be responsible for initial discovery and status updates among the cooperating VMs.

### 4.2.4. Dynamic Model

The dynamic model of the system is captured by the sequence diagrams in Appendix D.

First, cloud users may install or stop the XMPP server. The installation configures the XMPP server that will bootstrap the VPN network, which is used for initialize communication among the virtual machines. A user may then joins the VMs that will be nodes on the P2P network. Lastly, a user may leave the P2P network. In addition, administrators may, at any given time start, stop, restart the service.

# Glossary

API - Application Programming Interface: specifies how some software components should interact with each other [16].

Cloud computing: this phrase commonly refers to network-based services, which appear to be provided by real server hardware, and are in fact served up by virtual hardware that is simulated by software running on one or more physical machines [17].

Co-residency: a VM is co-resident with any VM when they are running on the same physical machine, and describes a great security risk when hosting VMs with sensitive data.

IP Address - Internet Protocol address: a numerical label assigned to each device participating in a computer network that uses the Internet Protocol for communication [19].

Hardware virtualization: this term refers to the creation of a virtual machine that acts like a real computer running an operating system [26].

LAN - Local Area Network: a computer network that interconnects computers in a limited area such as a home or a school using network media [21].

Live virtual machine migration: the process of moving a running virtual machine from a physical host to another physical host [6].

MTD - Moving-Target Defense: an approach that has been proposed to better protect important network systems and critical computing infrastructure by dynamically changing properties of their configuration in some way [1].

Non-live virtual machine migration: the process of moving a powered off virtual machine from a physical host to another physical host. After the transfer completes, the migrated virtual machine is restarted [6].

P2P network - Peer-to-peer network: a type of decentralized and distributed network architecture in which individual nodes in the network (called "peers") act as both suppliers and consumers of resources [22].

Private cloud: it is a cloud infrastructure operated solely for a single organization. It can be managed internally or by a third-party, and hosted internally or externally [17].

Public cloud: a cloud is said to be public when the services are rendered over a network that is open for public use, most commonly the Internet [17].

Side-channel attack: any attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms [23].

UML - Unified Modeling language: a standardized, general-purpose modeling language in the field of software engineering. It includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems [25].

Virtual cluster: a group of VMs configured for a common purpose with associated storage resource, operating system, software environment, communication protocol, and network configuration [12].

VM - Virtual Machine: a software-based emulation of a physical computer [27].

VMM - Virtual Machine Monitor or hypervisor: a piece of computer software, firmware, or hardware that creates and runs virtual machines [20].

Virtual network: a computer network that consists of virtual network links as opposed to physical (wired or wireless) links between connected devices. It is implemented using methods of network virtualization [28].

VPN - Virtual Private Network: A VPN extends a private network across a public network, such as the Internet. It enables a computer to send and receive data across shared or public networks as if it were directly connected to the private network, while benefiting from the functionality, security and management policies of the private network [29].

# Appendix

## 6.1 Appendix A - Complete use cases

### **Use Case – Configure XMPP Server.**

Details:

* Actor: Administrator
* Preconditions:
  + A dedicated XMPP server with access to the internet. A host or virtual machine.
  + The actor is logged into the XMPP server and has super user level privileges on that account.
* Description:
  + The use case begins when the actor runs the xmpp\_setup.py script on the command line of the dedicated XMPP server.
    - The actor must specify the username and password of the first admin user to be created by the script.
      * --user <user name>
      * --password <password>
    - The system will download and install the appropriate packages and then alter the configuration files required to successfully install the XMPP server.
    - The system shall create the admin user with the specified username and password.
    - The use case ends when the system returns a successful message for all requirements.
* Post Conditions:
  + The XMPP Server is setup and running.
  + The Administrator User is created on the XMPP Server.
  + The system returns a success message.

Alternative Courses of Action:

* None.

Exceptions:

* The system fails to download the required packages.
* The system does not recognize the arguments passed by the user.
* The system fails to properly install the packages required.

Related Use Cases:

* Configure VPN.

Decision Support

Criticality: High. This service is critical for the execution and implementation of the system. Without this service the VPN network is not feasible.

Risk: Low. The required packages are open-source and easily downloaded and installed. Documentation and configuration examples are readily available.

Constraints:

* Usability
  + The system shall provide a help menu for each available option that includes a description and example.
* Performance
  + The system shall complete the setup in less than one minute. Not including the time it takes to download the required packages as this is highly variable.
* Supportability
  + Any POSIX-compliant OS will be able to support this function.
* Implementation
  + This functionality will be implemented in its own setup script which will run in the command line. It will be written in Python. This functionality maybe called up by other functions.

Owner: Francois D’Ugard

Initiation date: 10/13/2014

Date last modified: 10/13/2014

### **Use Case – Configure VPN Node.**

Details:

* Actor: Administrator
* Preconditions:
  + A dedicated XMPP server with access to the internet. A host or virtual machine.
  + The XMPP server is running successfully
  + The actor is logged into the VM to be configured as the node.
  + The VPN node is accessible to the XMPP server vice versa.
* Description:
  + The use case begins when the actor runs the vpn\_node\_setup.py script on the command line of the VM.
    - The actor must specify the username and password of the user to be created by the script.
      * --xmpp-user <user name>
      * --xmpp-password <password>
    - The actor must specify the address of the VPN node, the network mask, the IP address of the XMPP server.
      * --ipop-address <ip address assigned to this VM on the VPN>
      * --ipop-mask <network mask>
      * --xmpp-host <IP address of the XMPP server>
    - The system will download and install IPOP then create the configuration file with the given arguments.
    - The system shall create the XMPP user with the specified username and password, via RPC call to the XMPP server.
    - The use case ends when the system returns a successful message for all requirements.
* Post Conditions:
  + The required configuration files have been created on the VM.
  + The User is created on the XMPP Server.
  + The system returns a success message.

Alternative Courses of Action:

* None.

Exceptions:

* The system fails to download the required packages.
* The system does not recognize the arguments passed by the user.
* The system fails to properly install the packages required.
* The system fails to write the configuration file.

Related Use Cases:

* Setup XMPP server.
* Join VPN
* Leave VPN

Decision Support

Criticality: High. This service is critical for the execution and implementation of the system. Without this service the VPN network is not feasible.

Risk: Low. The required packages are open-source and easily downloaded and installed. Documentation and configuration examples are readily available.

Constraints:

* Usability
  + The system shall provide a help menu for each available option that includes a description and example.
* Performance
  + The system shall complete the setup in less than one minute. Not including the time it takes to download the required packages as this is highly variable.
* Supportability
  + Any POSIX-compliant OS will be able to support this function.
* Implementation
  + This functionality will be implemented in its own setup script which will run in the command line. It will be written in Python. This functionality maybe called up by other functions.

Owner: Francois D’Ugard

Initiation date: 10/13/2014

Date last modified: 10/13/2014

### **Use Case – Join VPN.**

Details:

* Actor: Administrator
* Preconditions:
  + A dedicated XMPP server with access to the internet. A host or virtual machine.
  + The XMPP server is running successfully
  + The actor is logged into the VM to be joined into the VPN.
  + The VPN node is accessible to the XMPP server vice versa.
  + There is a correct configuration file available to the function.
* Description:
  + The use case begins when the actor runs the join\_vpn.py script on the command line of the VM.
    - The actor must specify the configuration file.
      * --conf <configuration file>
    - The system will start the ipop-tincan program with the MCCVPN controller.
    - The use case ends when the system returns a successful message for all requirements.
* Post Conditions:
  + There exists a new network interface with the IP and Mask specified in the configuration file.
  + The VM can reach other VMs through the VPN
  + The system returns a success message.

Alternative Courses of Action:

* None.

Exceptions:

* The system does not recognize the arguments passed by the user.
* The system is unable to read or find the specified configuration file.
* The system fails to start the ipop-tincan or MCCVPN controller.

Related Use Cases:

* Setup XMPP server.
* Leave VPN

Decision Support

Criticality: High. This service is critical for the execution and implementation of the system. Without this service the VPN network is not feasible.

Risk: Low. The required packages are open-source and easily downloaded and installed. Documentation and configuration examples are readily available.

Constraints:

* Usability
  + The system shall provide a help menu for each available option that includes a description and example.
* Performance
  + The system shall complete the setup in less than one minute. Not including the time it takes to download the required packages as this is highly variable.
* Supportability
  + Any POSIX-compliant OS will be able to support this function.
* Implementation
  + This functionality will be implemented in its own setup script which will run in the command line. It will be written in Python. This functionality maybe called up by other functions.

Owner: Francois D’Ugard

Initiation date: 10/13/2014

Date last modified: 10/13/2014

### **Use Case – Leave VPN.**

Details:

* Actor: Administrator
* Preconditions:
  + A dedicated XMPP server with access to the internet. A host or virtual machine.
  + The XMPP server is running successfully
  + The actor is logged into the VM to be removed from the VPN.
  + The VM is actually joined to the VPN
  + The VPN node is accessible to the XMPP server vice versa.
* Description:
  + The use case begins when the actor runs the leave\_vpn.py script on the command line of the VM.
    - The system will stop the ipop-tincan program and the MCCVPN controller.
    - The use case ends when the system returns a successful message for the removal of the VM from the VPN network.
* Post Conditions:
  + The IPOP network interface no longer exists on the system.
  + The VM cannot reach other VMs through the VPN
  + The system returns a success message.

Alternative Courses of Action:

* None.

Exceptions:

* The system does not recognize the arguments passed by the user.
* The system fails to stop the ipop-tincan or MCCVPN controller.

Related Use Cases:

* Setup XMPP server.
* Join VPN

Decision Support

Criticality: High. This service is critical for the execution and implementation of the system. Without this service the VPN network is not feasible.

Risk: Low. The required packages are open-source and easily downloaded and installed. Documentation and configuration examples are readily available.

Constraints:

* Usability
  + The system shall provide a help menu for each available option that includes a description and example.
* Performance
  + The system shall complete the setup in less than one minute. Not including the time it takes to download the required packages as this is highly variable.
* Supportability
  + Any POSIX-compliant OS will be able to support this function.
* Implementation
  + This functionality will be implemented in its own setup script which will run in the command line. It will be written in Python. This functionality maybe called up by other functions.

Owner: Francois D’Ugard

Initiation date: 10/13/2014

Date last modified: 10/13/2014

### **Use Case – Start XMPP Server.**

Details:

* Actor: Administrator
* Preconditions:
  + A dedicated XMPP server with access to the internet. Installed on a host or virtual machine.
  + The actor is logged into the XMPP server and has super user level privileges on that account.
  + A correct and complete configuration file is located on the server that is to function as the XMPP server.
* Description:
  + The use case begins when the actor runs the xmpp\_controller.py script on the command line of the dedicated XMPP server.
    - The actor must specify the usage parameter accepted by the script.
      * --start
    - The system will start the XMPP server following the parameters included within the configuration file.
    - The use case ends when the system returns a successful message for all requirements.
* Post Conditions:
  + The XMPP Server is started.
  + The system returns a success message.

Alternative Courses of Action:

* None.

Exceptions:

* The configuration file either incorrect or incomplete or missing.
* The system does not recognize the arguments passed by the user.

Related Use Cases:

* Configure VPN.

Decision Support

Criticality: High. This service is critical for the execution and implementation of the system. Without this service the VPN network is not feasible.

Risk: Low. The required packages are open-source and easily downloaded and installed. Documentation and configuration examples are readily available.

Constraints:

* Usability
  + The system shall provide a help menu for each available option that includes a description and example.
* Performance
  + The system shall complete the setup in less than one minute. Not including the time it takes to download the required packages as this is highly variable.
* Supportability
  + Any POSIX-compliant OS will be able to support this function.
* Implementation
  + This functionality will be implemented in its own setup script which will run in the command line. It will be written in Python. This functionality maybe called up by other functions.

Owner: Francois D’Ugard

Initiation date: 10/15/2014

Date last modified: 10/15/2014

### **Use Case – Stop XMPP Server.**

Details:

* Actor: Administrator
* Preconditions:
  + A dedicated XMPP server with access to the internet. Installed on a host or virtual machine.
  + The actor is logged into the XMPP server and has super user level privileges on that account.
  + The XMPP Service is running.
* Description:
  + The use case begins when the actor runs the xmpp\_controller.py script on the command line of the dedicated XMPP server.
    - The actor must specify the usage parameter accepted by the script.
      * --stop
    - The system shall store the state of all connections, virtual machines, paths, and VPN groups in storage.
    - The system will stop the XMPP server.
    - The use case ends when the system returns a successful message indicating that the server has been stopped successfully.
* Post Conditions:
  + The state of the objects of the system are written to disk.
  + The XMPP Server is stopped.
  + The system returns a success message.

Alternative Courses of Action:

* None.

Exceptions:

* The XMPP Server is not running.
* The XMPP Server is unable to connect to the database.
* The system does not recognize the arguments passed by the user.

Related Use Cases:

* Configure VPN.

Decision Support

Criticality: High. This service is critical for the execution and implementation of the system. Without this service the VPN network is not feasible.

Risk: Low. The required packages are open-source and easily downloaded and installed. Documentation and configuration examples are readily available.

Constraints:

* Usability
  + The system shall provide a help menu for each available option that includes a description and example.
* Performance
  + The system shall complete the setup in less than one minute. Not including the time it takes to download the required packages as this is highly variable.
* Supportability
  + Any POSIX-compliant OS will be able to support this function.
* Implementation
  + This functionality will be implemented in its own setup script which will run in the command line. It will be written in Python. This functionality maybe called up by other functions.

Owner: Francois D’Ugard

Initiation date: 10/15/2014

Date last modified: 10/15/2014

### **Use Case – Restart XMPP Server.**

Details:

* Actor: Administrator
* Preconditions:
  + A dedicated XMPP server with access to the internet. Installed on a host or virtual machine.
  + The actor is logged into the XMPP server and has super user level privileges on that account.
  + The XMPP Service is running.
* Description:
  + The use case begins when the actor runs the xmpp\_controller.py script on the command line of the dedicated XMPP server.
    - The actor must specify the usage parameter accepted by the script.
      * --restart
    - The system shall store the state of all connections, virtual machines, paths, and VPN groups in storage.
    - The system will stop the XMPP server.
    - The system will start the XMPP server.
    - The use case ends when the system returns a successful message indicating that the server has been stopped successfully.
* Post Conditions:
  + The last state of objects within the system are written to disk.
  + The XMPP Server is started.
  + The system returns a success message.

Alternative Courses of Action:

* None.

Exceptions:

* The XMPP Server is not running.
* The XMPP Server is unable to connect to the database.
* The system does not recognize the arguments passed by the user.

Related Use Cases:

* Configure VPN.

Decision Support

Criticality: High. This service is critical for the execution and implementation of the system. Without this service the VPN network is not feasible.

Risk: Low. The required packages are open-source and easily downloaded and installed. Documentation and configuration examples are readily available.

Constraints:

* Usability
  + The system shall provide a help menu for each available option that includes a description and example.
* Performance
  + The system shall complete the setup in less than one minute. Not including the time it takes to download the required packages as this is highly variable.
* Supportability
  + Any POSIX-compliant OS will be able to support this function.
* Implementation
  + This functionality will be implemented in its own setup script which will run in the command line. It will be written in Python. This functionality maybe called up by other functions.

Owner: Francois D’Ugard

Initiation date: 10/15/2014

Date last modified: 10/15/2014

## 6.2 Appendix B - Use Case Diagrams

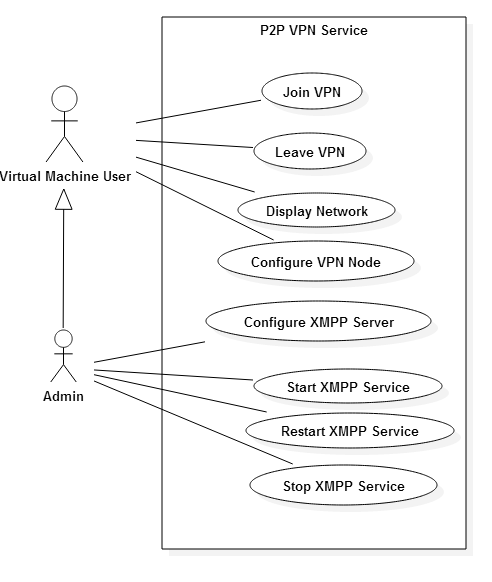


Figure 6.1 - Use Case Diagram

## 6.3 Appendix C - Static UML diagrams

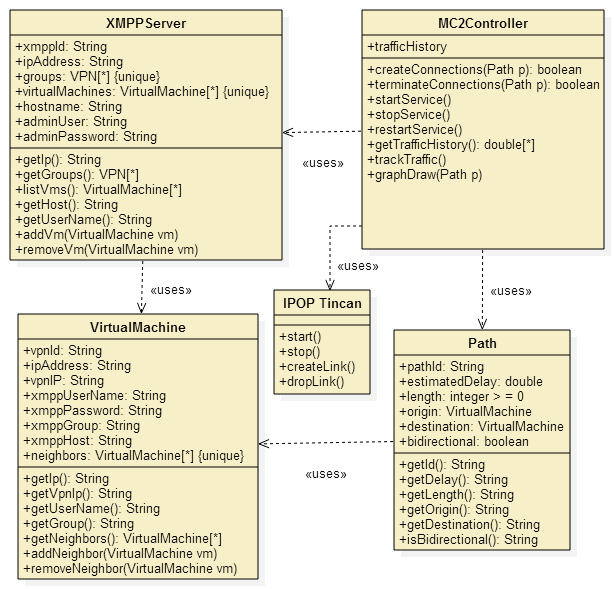


Figure 6.2 - Class Diagram

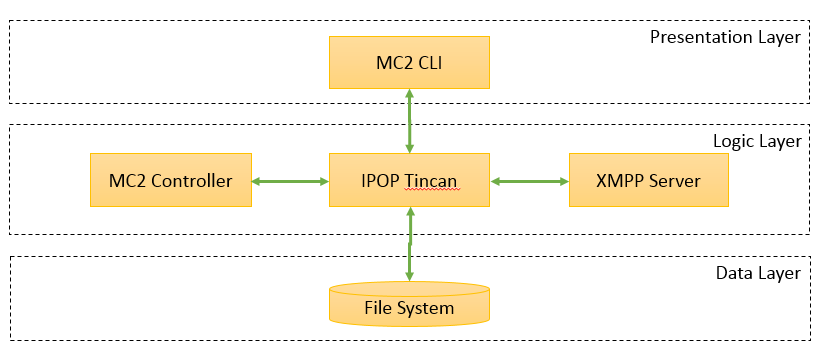


Figure 6.3 - Three Tier Architecture

## 6.4 Appendix D - Dynamic UML Diagrams

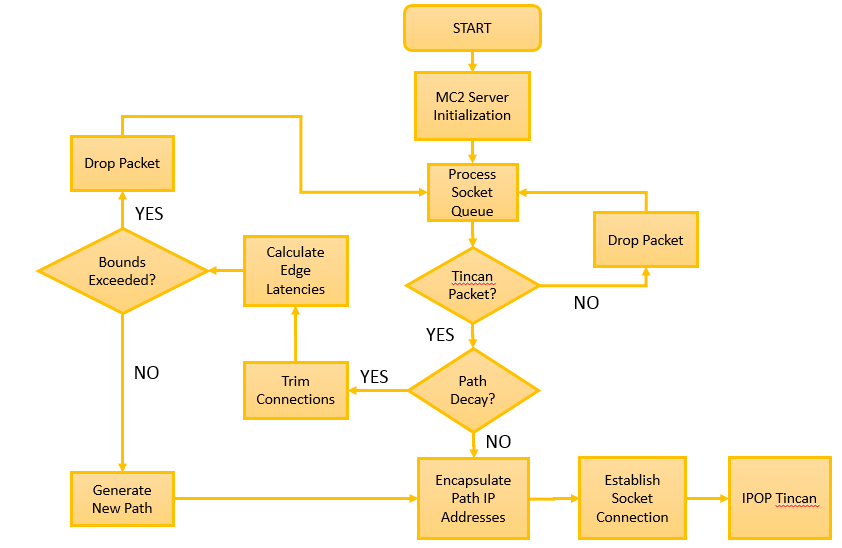


Figure 6.4 - State Chart

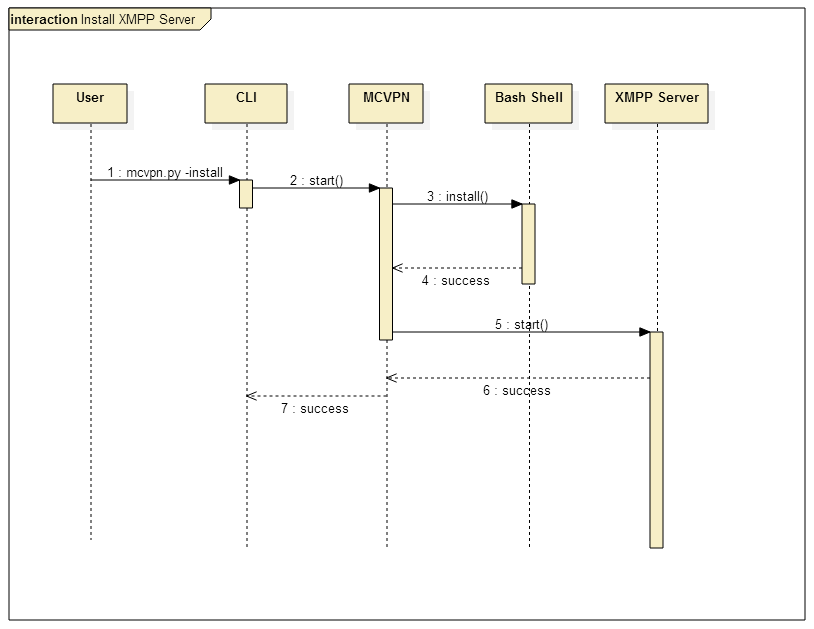


Figure 6.5 - Install XMPP Server

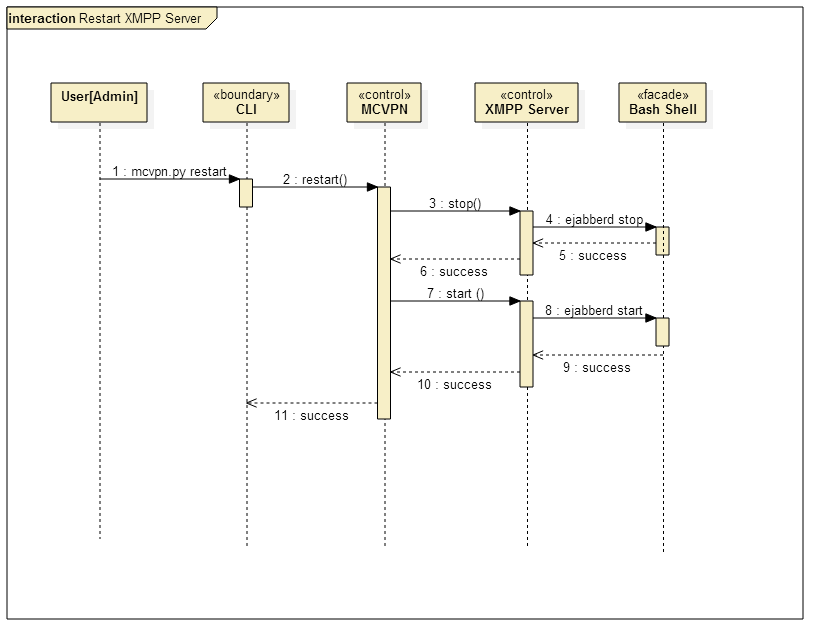


Figure 6.6 - Restart XMPP Server

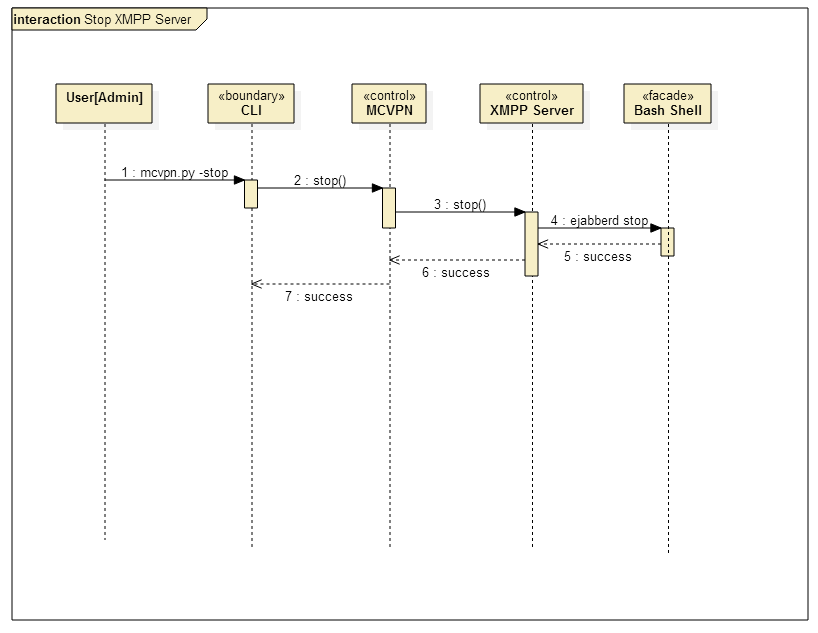


Figure 6.7 - Stop XMPP Server

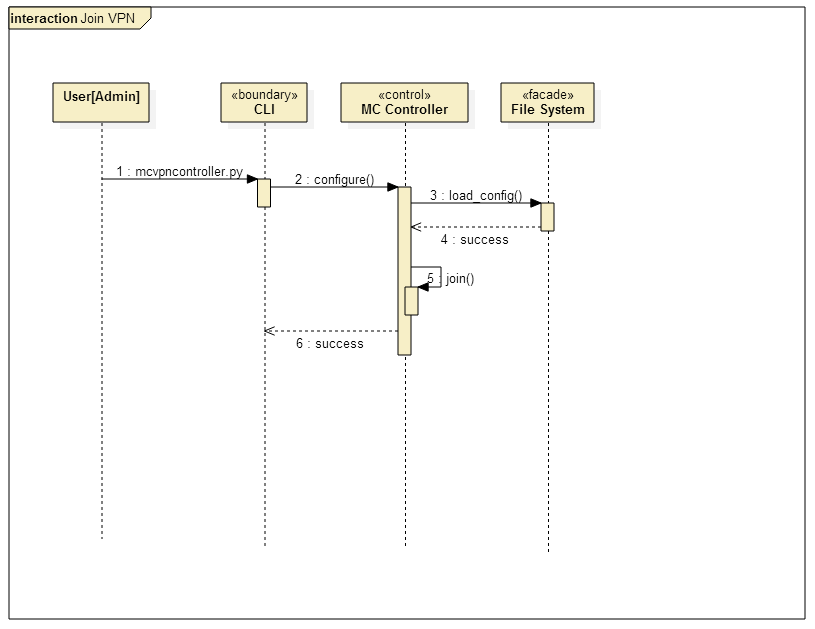


Figure 6.8 - Join VPN

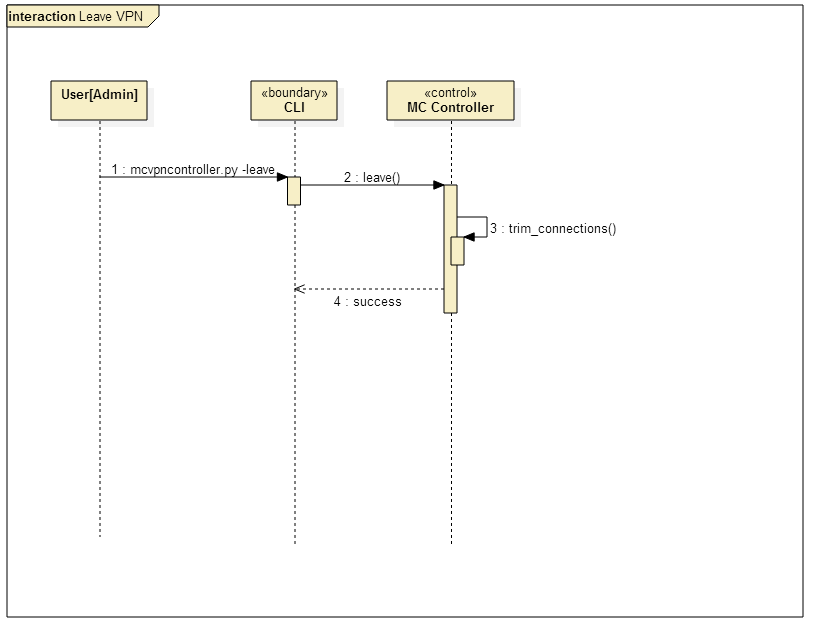


Figure 6.9 - Leave VPN

## 6.5 Appendix E - User Interface Designs.

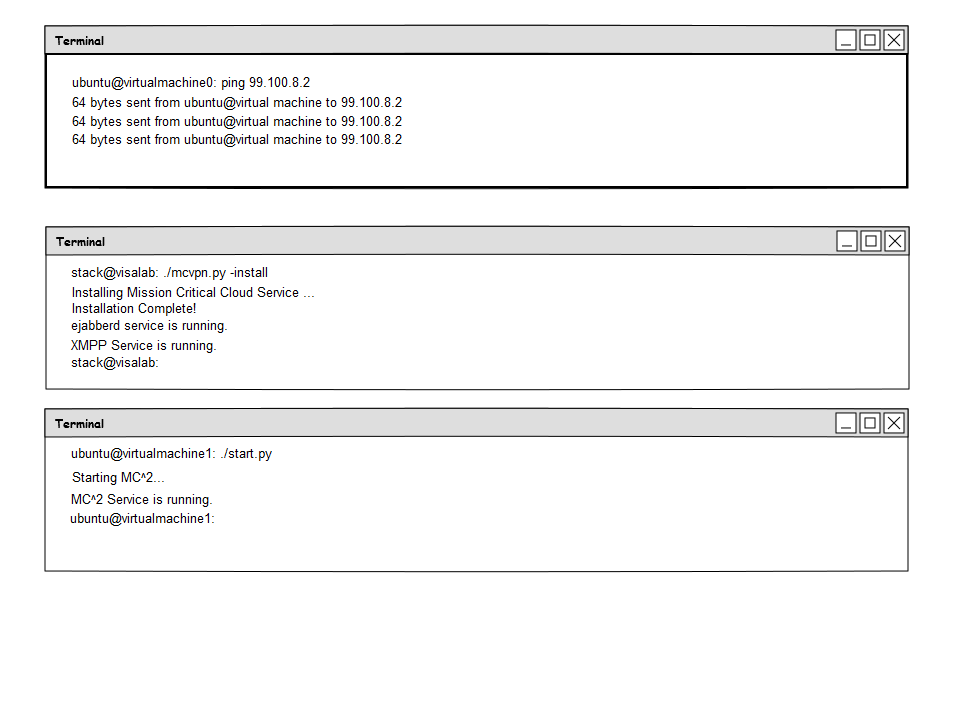


Figure 6.10 - User Interface Mockups

## 6.6 Appendix F - Diary of meeting and tasks.

9/05/2014

Present: Dr. Ming Zhao, Francois D’Ugard

Location: ECS 363

Time: 3:00 PM - 3:50 PM

Highlights:

* Discussed Current status of project
* Discussed challenges regarding OpenStack Configuration

9/15/2014

Present: Dr. Ming Zhao, Francois D’Ugard

Location: ECS 363

Time: 12:00 PM - 12:50 PM

Highlights:

* Discussed Current status of project
* Discussed differences between previous project and this semester's senior project.
* Dr. Zhao went into fine detail what differentiates my contributions from the last senior project.
* Z elaborated on the requirements of the new IPOP controller for this iteration of the senior project
* Maintains IPOP sparse network connections to hide or mask controller server
* Controller should use DHT or other algorithm to route messages via existing IPOP VPN links
* Demo application must use the VPN links to route messages among the system.

11/03/2014

Present: Dr. Ming Zhao, Francois D’Ugard

Location: ECS 363

Time: 11:00 AM - 12:00 PM

Highlights:

* Discussed Current status of project
* Dr. Zhao said he would contact Dr. Figueiredo to discuss implementation of message forwarding
* Discussed implementation of message forwarding functionality
* Mentioned encapsulation idea
* Discussed demonstration

# 7. References

[1] Carvalho, Marco, Jeffrey M. Bradshaw, et al. "Command and Control Requirements for Moving-Target Defense."IEEE. 1541-1672/12. (2012): 79-85. Print.

[2] Deshpande, Umesh, Xiaoshuang Wang, et al. "Live Gang Migration of Virtual Machines." International Symposium on High-Performance Parallel and Distributed Computing. (2011): 135-46. Print.

[3] Ganguly, Arijit, et al. “IP over P2P: Enabling Self-configuring Virtual IP Networks for Grid Computing,” Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International. IEEE, 2006.

[4] Haadi Jafarian, Jafar, Ehab Al-Shaer, and Qi Duan. "OpenFlow Random Host Mutation: Transparent Moving Target Defense using Software Defined Networking." ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. (2012): 127-32. Print.

[5] Ristenpart, Thomas, Eran Tromer, et al. "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds." ACM Conference on Computer and Communications Security. (2009): n. page. Print.

[6] Yackoski, Justin, and Scott A. DeLoach. "Mission-oriented Moving Target Defense on Cryptographically Strong Network Dynamics." Annual Cyber Security and Information Intelligence Research Workshop. (2011): n. page. Print.

[7] Zhuang, Rui, Su Zhang, et al. "Investigating the Application of Moving Target Defenses to Network Security." 6th International Symposium on Resilient Control Systems. (2013): 162-69. Print.

[8] Zhuang, Rui, Su Zhang, et al. "Simulation-based Approaches to Studying Effectiveness of Moving-Target Network Defense." National Symposium on Moving Target Research. (2012): n. page. Print.

[9] Wikipedia contributors. "Application programming interface." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 24 Jan. 2014. Web. 25 Jan. 2014.

[10] Wikipedia contributors. "Cloud computing." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 17 Jan. 2014. Web. 21 Jan. 2014.

[11] Wikipedia contributors. "IP address." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 21 Jan. 2014. Web. 22 Jan. 2014.

[12] Wikipedia contributors. "Hypervisor." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 7 Jan. 2014. Web. 21 Jan. 2014.

[13] Wikipedia contributors. "Local area network." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 16 Jan. 2014. Web. 21 Jan. 2014.

[14] Wikipedia contributors. "Peer-to-peer." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 13 Jan. 2014. Web. 21 Jan. 2014.