Implementation and Experimentation with OpenStack Cloud Computing System

Introduction:
Cloud Computing Technology (CCT) can exist in many forms and is fundamentally a complex system-of-systems. In PART II of this report, we proved theoretically through a fault tree analysis that users of cloud computing systems are more at risk than users of non-cloud computing systems, given similar system configurations. In PART III, we demonstrate the above conclusion with a physical mini-cloud computing system. Our research approach consists of two parts: 1) building a physical mini-cloud system; 2) designing experiments on the mini-cloud system to explore vulnerabilities and test intrusion methods unique to cloud computing systems. Using a mini-cloud computing system as a case study allows us to apply general conclusions on the risk to CCT associated with a specific software/hardware system that is common in industry. We implemented a small-scale private cloud computing system based on OpenStack as a case study for this project.

The experimental cloud computing system is used as a case study to understand security risks of more general cloud computing systems. From an experimental perspective, we implemented a cloud cluster in our Phantom Systems Modeling Laboratory (PSML), with appropriate security software. The PSM Lab at UVA currently has six machines, on which many more virtual machines may be deployed. These PSM Lab machines have the ability to be supplemented with other desktop machines at the center.

The philosophy behind PSM is that multiple perspectives are needed to model a complex system. In this case, the multiple perspectives are captured by reproducing scenarios of attacks from different sources and by different methods, with varying configurations of the OpenStack architecture and security protocols. The results of this experimentation
process demonstrate the differences between the security risks to CCT and to non-CCT systems.

System Description

1. System Architecture

The purpose of building our own cloud was to explore vulnerabilities and develop hacking and penetration approaches specific to cloud computing systems, which is not appropriate or legal using a public environment. Also, building our own cloud allows us to become more familiar with the underlying hardware, software, and processes of a cloud computing system. We chose OpenStack because it was freely available and is a good representation of systems used in practice.

OpenStack is an operating system for Infrastructure-as-a-Service (IaaS) clouds, providing Virtual Machine (VM) instances for users. A VM instance is composed of an image (which determines the configuration), a flavor (which determines the size), and (sometimes) a persistent storage block. Currently our cloud is using basic open source images. Nova and Glance can use the same VM image format (.ami files) as that used by Amazon’s Elastic Compute Cloud (EC2). OpenStack supports two forms for storage: objects with Swift, and volumes (blocks) with Cinder. Volumes are persistent storage allocated to a VM instance such that the virtual machine views it as a local disk. Alternatively, object files are viewed by the VM instance using them as being on some remote disk. As an IaaS system, OpenStack VM instances are often configured and run servers in production environments. Our implementation is currently running basic open-source applications on individual VM instances for demonstration purposes.

2. Cloud Hardware

The CCT system developed for this project is comprised of four computer nodes and one controller. The entire cloud currently runs on the “Folsom” release of Open Stack. Each node runs the Kernel-based Virtual Machine (KVM) hypervisor. The cloud system is accessible through public IP via the Ubuntu OpenStack Dashboard, Horizon, as well as through standard secure shell (SSH) connection. In addition the entirety of the cloud is connected through a secondary set of Ethernet connections creating a separate private cloud. Upon connecting to the cloud system, users are able to load virtual machines from various images with a variety of formats. While not all of these selections are available to the public cloud, they are all available to the private cloud. The cloud system does not support persistent storage. As an alternative to persistent storage, users are allowed to save snapshots of current images, then load into those snapshots much the same as a user would load into an image.

The cloud system described above consists multiple interdependent subsystems, including user, cloud infrastructure (storage, computing, and network) and hypervisor, security, and cloud provisioning and management. These subsystems constitute a complex interconnected system of systems, which implies, from the Phantom System Models theory, that they share states. Furthermore, two subsystems that share one or
more states are more vulnerable to the same threat than a system that has no shared states, and thus they are more at risk, because an intruder would have more than one path to penetrate the CCT system of systems.

3. Cloud Software

OpenStack software is open-source software that is developed, maintained, and managed by the OpenStack Foundation, which is composed of a multitude of large and small corporations in the cloud industry. The version of OpenStack that we are using is called “Folsom.”

![Figure 8: Schematic of OpenStack Software Functionality](image)

OpenStack has a number of key software modules, referred to as “services.” The following is the Nova administration manual description of each service:

- Object Store (codenamed "Swift") provides object storage. We have not used any object storage in our cloud, though it would be possible to do so.

- “Compute (codenamed "Nova") provides virtual servers upon demand. What the documentation refers to as “servers” are just virtual machines, which do not need to function as servers in the strict sense.

- Dashboard (codenamed "Horizon") For our cloud, this interface can be accessed by directing any browser to 128.143.56.140/horizon, allowing users to launch instances from the web.

- “Identity (codenamed "Keystone") provides authentication and authorization for all the OpenStack services. It also provides a service catalog of services within a particular OpenStack cloud.” This manages both human users and services, which it treats like human users. Keystone characterizes identity in three ways:
  - User – this is the most basic form of identity, and currently there is a user designated for each service as well as “demo” and “admin” users.
- Role – each user can take on multiple roles, but roles are handled automatically and are not noticeable in normal operations.
- Tenant – each user belongs to one or more tenants. The current tenants are: demo, invisible_to_admin, service, and admin. Because every instance exists within a particular tenant (which was formerly referred to as a “project”), a tenant must be selected on the dashboard when working with instances, as shown in Figure 2. The tenants demo and admin are the only ones that are applicable to us, and demo is the only one we have actually been using.

Figure 9: Dashboard screenshot indicating where tenant (project) is selected

Figure 10 provides a visual explanation of these software modules and how they interact.
Vulnerabilities and Intrusion Methods Unique to CCT

One of the major difficulties in identifying weaknesses in cloud computing security is finding a form of attack that is unique to cloud computing. That is, almost any attack on a cloud system has some analog in conventional computing. For example, VM images with a public IP address can be hacked using the same tools and techniques used against conventional hosts. However, many cases cannot be evaluated in such a clear way, since the definition of cloud computing itself is only expressed in loose qualitative terms. For the purposes of experimentation, however, two weaknesses that can be considered germane to cloud computing are those found in cloud management consoles and VM images.

Cloud management consoles: Since one of NIST’s key characteristics of cloud computing is “Broad Network Access,” a cloud by definition can be controlled remotely through some form of client, which is often browser-based. These management consoles constitute a significant source of security risk because attackers can take advantage of common weaknesses such as cross-site scripting (XSS) and cross-site request forgery (CSRF) to gain access to a victim’s cloud. This problem is exacerbated by the increasing use of mobile devices and the so-called “Bring Your Own Device” (BYOD) movement. With more disparate points of access to the cloud, there are more opportunities for the attackers to find security flaws and for legitimate users to make mistakes in managing the
security of their system. For example, the security tokens used to access a VNC console for a VM instance are, by default, sent in plain text, that is, unencrypted. Anyone on the network who captures the appropriate packet will then be able to access the VNC console login page, bypassing the normal authentication required for OpenStack Horizon. Though remote access is used in conventional computing environments, this source of risk can be considered to be germane because of how fundamental it is to cloud computing.

VM images: Because images are the configuration templates from which virtual machines are created, their security and integrity is critical. However, many images come from open-source communities, where attackers can potentially make malicious modifications. Furthermore, the images need not be modified maliciously to be dangerous, because benign users and creators of images can make mistakes that leave security flaws open for attackers to exploit. The management consoles can be considered a source of risk that is unique to cloud computing, because the scale and criticality of its use (even though some virtualization), is used in conventional computing systems.

Other potential areas of exploration would be weaknesses in multi-tenancy, data reminisce, cryptographic protocol implementation, cloud auditing, inter-VM side channels, and many others.

We have developed and experimented with ways with which to demonstrate exploitation of the weaknesses in the cloud management consoles mentioned above in our own cloud. Among the many challenges of this process is the task of identifying the specific chain (or chains) of events that will contribute to the final event of interest. For this final event, we have decided to focus on loss of confidentiality, which by its nature, must be coupled with consideration of the loss of integrity. It should be noted that it would be equally worthwhile to consider risk of denial of service (i.e., loss of availability) in the cloud. This process is facilitated by the wide availability of penetration-testing software. For example, we experimented with a software package called WireShark for some web management console attacks.

**Exploring And Experimenting Potential Security Issues Of Cloud Computing Technology**

Our current research focused on exploring potential security issues of CCT, especially in OpenStack, which is being used by our experiments.

The issue with VM images arises from the fact that every VM instance is created from an image, which essentially determines the operating system, applications, and overall configuration of the instance. Images can include a wide variety of security weaknesses both intentionally and accidentally introduced (Dhanjani et al. 2009). This is especially problematic since images are shared online and consequently have uncertain provenance. Through monitoring this transfer of images in setting up a VM instance on a given node, the potential exists for an attacker to capture these packets and reconstruct another user’s VM on their own machine.
The issue with VM images reduces the security of the cloud computing infrastructure because a user with illegal administrator privileges is able to collect valuable information and send it to a remote location, while only having access to a cloud computer during a single session. To demonstrate the possibility of this type of attack, we posit that someone with prohibited administrator access (the insiders) is able to install a program in a VM image that runs as a background process in a VM instance to collect and send information to a remote computer or email address. To implement and test this idea, we inject a key log and email program into a VM image that will run unnoticed on one of the cloud virtual environments. This background process will track all the keystrokes a user makes and will record that information in a text file. That text file will be emailed to a specific email address at timed intervals. We have taken a few approaches to try and achieve this objective of collecting and sending information to a remote computer. The first approach attempted to utilize a user created key log system from the Linux app center.

Furthermore, instead of trying to use a Linux app from the app store, we moved to a lower level key log program that is written in C code, which is compiled and capable to run on any machine. The C code is a collection of files stored in a single folder on the computer. In order to have the code log the keystrokes, first the code must be compiled, and then the program must be started. The program is able to log keys and it even runs in the background, which is a huge improvement over the last method (as demonstrated below). No matter what the user is doing, anything they type is automatically logged to a specific file.

Conclusions
Our implementation of a mini-cloud computing system enriched our knowledge about the architecture, hardware, and software of a typical cloud computing system and helped us to identify commonly shared resources among subsystems such that unique forms of cyber attack are explored and tested. We discussed potential vulnerabilities of two major types of these forms of cyber attack: cloud management consoles and VM images. Another implication of our implementation of the mini-CCT is that CCT is a mature technology with many commercial systems implemented by open source software and similar architecture. This implies that a potential hacker or intruder is able to find detailed technical resources and gain significant knowledge about a target system. This is another source of risk on CCT compared to other proprietary non-CCT systems.

Our experimentation of the mini-cloud computing system demonstrated that exploiting vulnerabilities in CCT is feasible and it takes different forms, although a certain level of knowledge of system structure, security protocol, and programming is required. A cloud user’s security cannot be ensured since the virtual machine provided by the cloud provider may have been compromised in the first place. A cloud user has to understand and have established protocols to safeguard this source of risk when deciding to choose between cloud computing and non-CCT proprietary system.