UPPSALA
UNIVERSITET

IT 18 002
Examensarbete 30 hp
Februari 2018

Implementing Security Rules, Safeguards, and IPS tools for Private Cloud Infrastructures

GROOT: Infrastructure Security as a Service (ISaaS)

Aleksander Okonski

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Implementing Security Rules, Safeguards, and IPS tools for Private Cloud Infrastructures

Aleksander Okonski

Clouds are a large part of todays computing space, with individu- als having the ability to crate their own cloud. The OpenStack cloud suite eases the deployment and management of cloud services. How- ever, security is one core area that is difficult to isolate and has to be addressed at each level, ranging from low-level system security to the user-facing multi-tenant environments. There are solutions available that offer end-to-end security but most of them are proprietary and with their sophisticated licensing scheme, expertise that might be af- fordable for large enterprises but difficult for medium and small-scale organizations is required. The aim of this project is to design a min- imalistic security service for the OpenStack environment that helps cloud administrators get first-hand information regarding any activi- ties that may cause threats to the instance or the whole tenant in the cloud infrastructure. The project created a proof of concept system that once deployed was able to detect potential misconfigurations and threats. The system was tested in a real world scenarios and proved to work, finding several machines that were launched without correct configurations.

Handledare: Salman Toor
Ämnesgranskare: Björn Victor
Examinator: Mats Daniels
IT 18 002
Tryckt av: Reprocentralen ITC

Contents

1 Introduction

Cloud computing is a term to describe computational resources that are located on a server running a software layer that disenfranchises the hardware and software. This allows for software to run independently of the hardware and allows a middle layer to control how the two interact. The hardware can then run several different software systems on the same hardware with little overhead. The servers running the cloud are typically hosted in a data center in an external location. Some examples of cloud computing are Amazon Web Services (AWS) [16] or Microsoft Azure [17]. Cloud computing is not a new phenomenon[46], it had its origins in the early 2000's [46]. The last several years, cloud computing has steadily grown (Figure 1) as a platform for businesses.

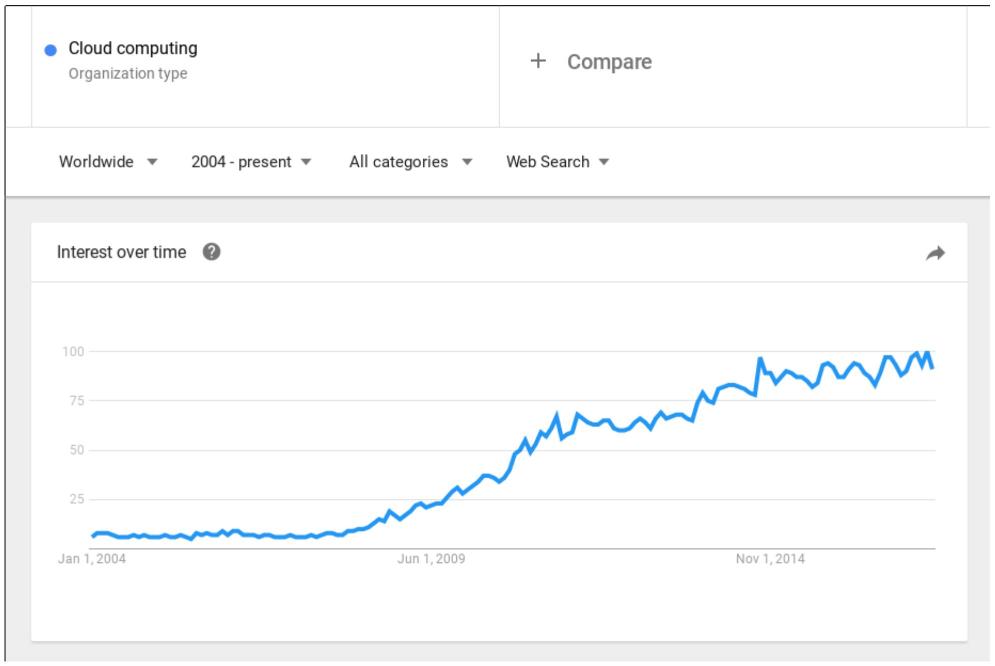


Figure 1: Cloud computing trend from Google Trends. Shows the interest in cloud computing between the years of 2004 to 2017. [4]

Cloud computing can be divided into 3 different categories: public, private, and hybrid. The two main categories are public and private with the hybrid a mixture of these. Public and private clouds are separated by two main categories, finances and physical location. Public clouds allow anyone to pay for the resources that they would like to use. Therefore a user can start up a virtual machine (VM) and then only pay for the amount of time that they use the resource. These VMs that a user starts up are usually hosted in data centers owned by the cloud provider. This means that the data the user puts onto the servers are stored on the cloud providers servers.

The large players in this space are Amazon AWS [16] and Microsoft Azure [17]. Private clouds typically only allow users of a particular organization to gain access to their resources. Once the users gain access they may still need to pay for usage. The private clouds are simply more restrictive on who gains access the cloud resource. In addition, private clouds are hosted at the organization's premise. This means that the servers hosting the cloud are located in the organization and use the organization's network. This allows users to start up VMs and have the data and computations happen on premise.

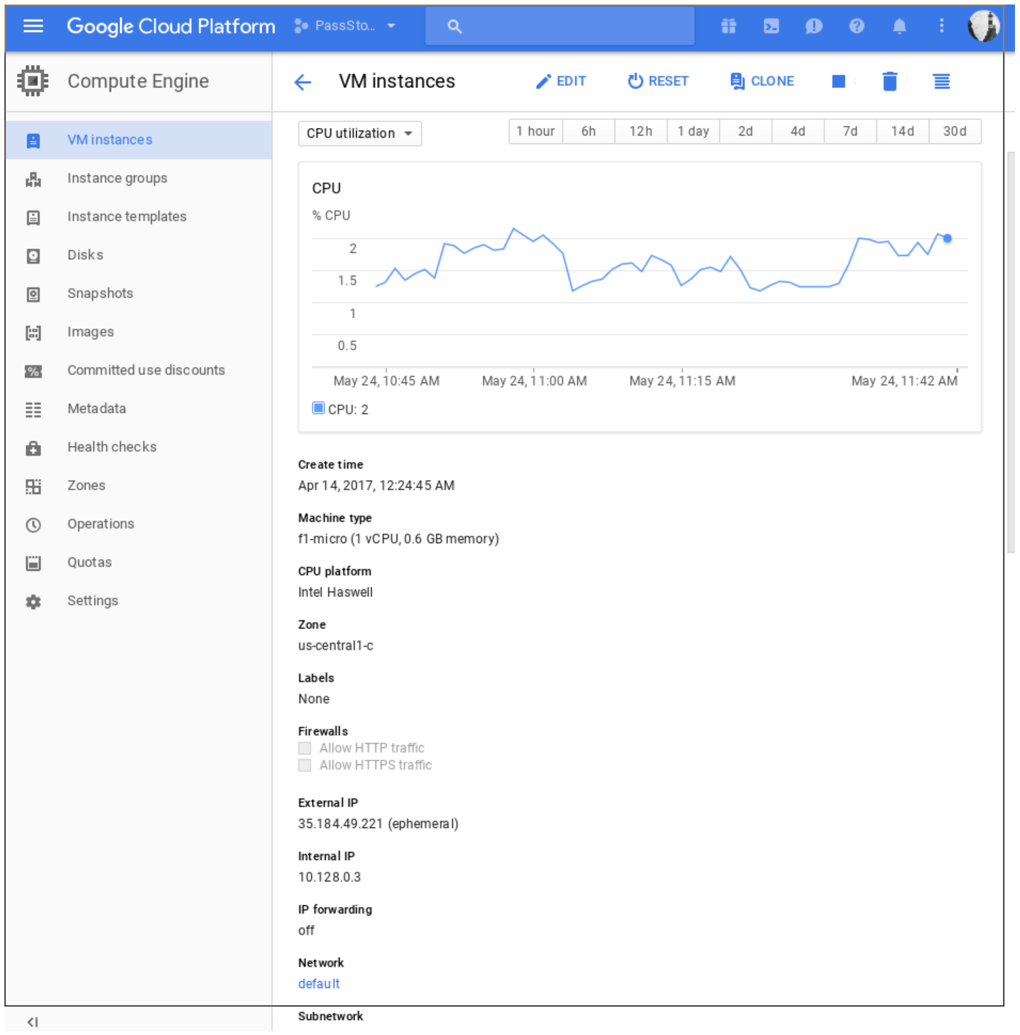
What the cloud provides to users varies depending on the feature set of the cloud and the user's needs. As a minimum, the cloud can typically create new virtual machines (VMs) that the user then has root access to. The users are then free to run any program they like. As cloud systems become more advanced with ever-expanding user needs, more and more companies and individuals are migrating over to the cloud[31]. This expansion leads to a larger possibility for cyber attacks.

Public attacks on computer systems are growing year by year [42]. Cloud systems and the VMs that run on top of them are starting to be a major target for cyber criminals [30]. The transition to the cloud brings about new challenges, especially in the security space involving cloud systems. The users of the cloud start up systems that have direct access to the Internet. These machines are susceptible to attacks especially with users not understanding the best practices in security [43]. Users who configure software poorly are at a greater risk of bugs and have a expanded attack surface that can then be compromised. Public cloud providers have created proprietary solutions to help mitigate their users' systems being compromised. However private clouds do not have the manpower or funds to be able to tackle such a threat.

The aim of this thesis is to facilitate security information in private cloud infrastructure when an organization does not have the resources to deploy sophisticated security solutions. Open source cloud solutions, such as OpenStack [7], are becoming sought after solutions that allow anyone to create there own private cloud. These implementations lack a clear-cut solution to security monitoring. This could lead to weaknesses being exploited in the private cloud allowing unauthorized users access to resources. To alleviate the threat a three-step solution was created. The first was to create an outline that new users could follow to understand basic security practices when connecting to a VM. The second was to create an active monitoring solution within multi-tenant environments. The solution is created to be versatile for any type of environment and for administrators to be able to obtain information with ease. Third, an information gathering tool was created to obtain system data before termination of the infected VM on the cloud system. The project is designed to approach the security for private clouds actively instead of passively. The administrators should be able to detect vulnerabilities early before they are exploited. The proposed framework is a prototype solution that is able to identify vulnerabilities and report to administrators. These results were able to be used by administrators to prevent the exploitation of the vulnerability.

2 Background

As can be seen in Figure 1, cloud computing paradigm has gained a lot of attention over the last several years. Business and academia are looking at solutions to migrate their existing infrastructure to the cloud. There are several reasons for this type of shift, most prominently: costs, scalability, and reliability [24]. The cloud offers some unprecedented advantages to a standardized computational model. One is able to pay for only the resources used, with more resources added/removed depending on the demand. This is more commonly referred to as the pay-as-you-go-model. Another advantage is the ability to start up/destroy several machines with little overhead. The usual use case for this is to scale up or down systems depending on the amount of work. Scaling can occur in two ways: vertical and horizontal scaling. Vertical scaling is when the VMs add more processing power, automatically adding resources such as CPU, memory, and hard drive space. As cloud systems are configurable they can have more cores, memory, and hard drive space allocated with no downtime. Horizontal scaling is when new machines are added to the cloud cluster to take on a larger workflow. Instead of adding more resources, like in vertical scaling, new machines are spun up with the pre-installed software pre-configured. These machines can then have some of the workload offloaded to them. Several companies are fronting the cloud revolution, some examples are Amazon[16], Google[21], Microsoft[17], and Digital Ocean[44]. These companies are providing a public cloud environment, that is to say, that anyone can pay for computer resources. A user needs to sign up, provide a credit card, and then start setting up and launching whatever configuration of a system they would like.



Create time
Apr 14, 2017, 12:24:45 AM

Machine type
f1-micro (1 vCPU, 0.6 GB memory)

CPU platform
Intel Haswell

Zone
us-central1-c

Labels
None

Firewalls
☐ Allow HTTP traffic
☐ Allow HTTPS traffic

External IP
35.184.49.221 (ephemeral)

Internal IP
10.128.0.3

IP forwarding
off

Network
default

Subnetwork

Figure 2: Google cloud interface for controlling VMs

The provider will then dedicate the specified hardware and allow the user to log in to that virtual machine and run whatever tasks the user would like to do. An example of the Google cloud compute engine is shown in

Figure 2. These public entities typically have teams dedicated to security and infrastructure monitoring. They have created proprietary solutions to ensure that their cloud environment [2] stay secure. As the security systems are a selling point for companies they are kept proprietary.

A private cloud is a different type of model where the cloud infrastructure is physically set inside a companies domain. Many companies would like to move over to a cloud-based model but do not want to migrate their private resources into the public domain. Therefore companies such as RackSpace, IBM, and VMware are providing private/hybrid cloud resources. This approach has part or the entire system built on site at the client's data center. This allows the client to have most of the benefits of the cloud but also keep the system within their network. These clients typically do not have the resources to create or manage a cloud security system.

2.1 Open Source vs Closed Source

An open source project is a type of project that has the source code to the project freely available for anyone to download, modify, and run. This allows a community of enthusiasts and commercial partners to support and expand the project. These projects can be very popular and can expand to a very large scale, some examples of popular projects are Linux [6], atom [5], and FreeBSD [3]. Users contribute time in writing code, filling out documentation, or filing bugs this slowly expands the project. Goals for projects are laid out by the community if a user does not agree with a decision they are allowed to fork a project and take it in a direction that they choose. Typical projects are funded through donations or through the time of users. Closed source coding projects are typically found in corporations. The code is not typically available and only the end product is distributed to consumers. The project goals are outlined to fit into a release time line. Full time workers then will work on the project to meet these deadlines. There typically is funding available for these types of projects with the end product monetized for profit.

2.2 Cloud Models

As mentioned above the cloud space consists of three types of clouds: public, hybrid, and private. The public cloud typically runs on a pay-as-you-go model. This means that anyone can sign up and pay for machine up time, storage, or network usage. The host machines that run the cloud environment are located in public data centers throughout the world. The virtual machines are created on these hosts and then access is given to them over the network. The users can then setup the environment to their liking. The public cloud has benefits of reliability, scaling, and flexibility. The private cloud consist of the host being located in a private hosting environment. These hosts typically exist on the organizations' premise and are connected to the internal network. The machines can then be reached from the internal network but cannot be reached from outside. The private cloud ensures that information stays within the organizations' network while allowing for scaling and flexibility. The hybrid environment merges the two, with having some of the machines located in private data centers and others located on public data centers. This allows an organization to have secure resources run in their private cloud while allowing the public cloud to enable rapid scaling. Each of these models have benefits and drawbacks and have to be evaluated for each individual project. The cloud models all share a similar set of tools employed to provide the necessary computational resources to users.

In the cloud computing space several computational models exist [41]. These models reflect the different components of a computer: hardware (Software as a Service), operating system (Platform as a Service), and software (Software as a Service).

Infrastructure is the basis of all systems, before cloud computing this was typically server and networking components. Infrastructure as a Service (IaaS) gives the users a basic virtual machine with the user needing to set up all programs and tools. This moves the responsibility of management of the hardware, network, and storage from the user to the operator. Microsoft gives a good explanation: "The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud physical infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components." [34] For the cloud, IaaS builds the backbone for the other modules to be built off of.

Platform as a Service (PaaS) gives the user the ability to create applications (web servers, databases, etc.) without the need to create the entire system from the ground up. The PaaS builds from the IaaS and utilizes the hardware of IaaS to provide the users the platform that they desire. Platform software is a layer of software that enables a user to build their own software from. Some examples of this are the Apache Web Framework[27], WordPress[51], and MySQL[40]. With the cloud model a user is now able to get access to platform software without having to deal with hardware or operating system.

Software as a Service (SaaS) allows for the user to utilize applications (Email, games, etc.) without the need to set up the underlying infrastructure. The cloud provider would be responsible ensuring that the infrastructure and OS are running correctly. The SaaS builds off of the last two layers to only provide a specific service for the user to use. A user then typically will use a web browser to access and utilize the software that is now run entirely on the cloud. The user can then preform work using the provided service with all of the underlaying configuration done automatically for them. Some examples of software are Office 365[35], Gmail[28], and Photoshop CC[14]. The user has no access to the underlying network, computer system, or storage.

2.3 Virtualization

At the most fundamental layer a cloud computer is a server running in a data-center that has a hypervisor which then allocates resources on demand. These hypervisores are the backbone of cloud computing allowing several virtual environments to use the same hardware. A hypervisor is a layer of code that sits between the computer components (bare metal) and the guest OS. There are two types of hypervisores that are present in systems they are named bare bone and hosted hypervisores. Figure 3 [1] shows these two in example format. These two hypervisors are very different in the way that they are created. A bare bones hypervisor acts like a small operating system which only manages the VMs while allowing most of the VMs to access hardware directly. A hosted hypervisor relies on an existing operating system to function. Although a hosted hypervisor can also directly allow VMs to communicate with the hardware without going through the OS. OpenStack uses the hosted hypervisor solution, having the administrators install the OpenStack platform on top of an existing operating system. The hypervisor then translates instructions from the guest system onto the hardware. There are several different hypervisors to choose from (Xen, Oracle VirtualBox, Oracle VM, KVM, VMware ESX/ESXi, or Hyper-V) with each having similar

[1]https://commons.wikimedia.org/wiki/File:Hyperviseur.png#/media/File:Hyperviseur.png

outcomes through different approach to the problem. To control the users and virtual machines many cloud providers (Amazon, Google, etc.) have created proprietary solutions. However, NASA and RackSpace Hosting [23] have created an open source version called OpenStack.

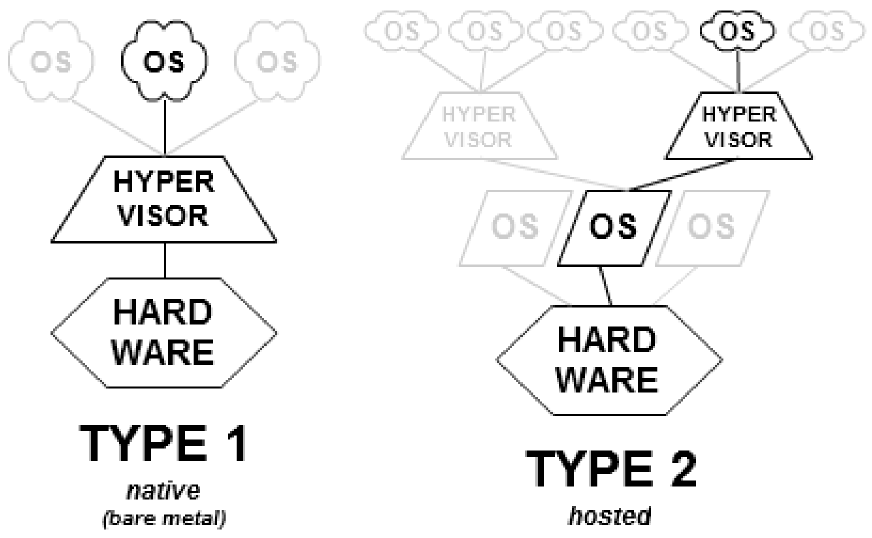


Figure 3: Different types of Hypervisors

2.4 Cloud Models

When cloud computing first started to take off, the main type of computing resource provided was IaaS in the public cloud [1]. This started to change in the recent years when two new types models for cloud computing emerged. Private and Hybrid clouds allowed for companies to utilize the power of the cloud while still having some or all of there resources located in their own data centers. The private cloud is where the data and computation is performed on the organizations own hardware. An example of this might be a business that would set up a cloud environment in their own data center. They could then allow users on their internal network to connect to the internal cloud resources. The hybrid model is a mixture between the public and private cloud. This is where some of the cloud functionally is located in a organizations internal infrastructure while some aspects are hosted in public clouds. An example of this could be a business that has development done on an internal cloud while having their production environment in the public cloud. To choose between the different cloud models is very dependent on the organization and the types of problems that the organization is attempting to solve.

2.5 Cloud Computing vs Standard Models

The cloud computing environment has been growing quickly the last several years [49]. However it is important to understand the similarities and differences that this new type of computing brings. At the core cloud computing is no different then older client server models. The data is still on hard drives, servers still use OS environments, and networks still connect the servers to the Internet. The main difference between the two models is that in the cloud you pay for what you use and hardware configurations/problems are not a concern. Some of the advantages that cloud computing posses is the fact that there is rapid scalability. It is possible to start up 100 machines and scale your service according to the users needs. There is also a level of abstraction between physical hardware and the systems. This provides a buffer, allowing for there to be problems with configurations, management, and physical hardware with little effect on the cloud hosted systems.

2.6 Threats to the Cloud

Cloud computing also has a different threat model compared to a normal dedicated server [52, 36, 32]. As cloud servers are, as the name suggests, in the cloud they usually will receive a pubic IP address that allows anyone to directly talk to them. This allows for direct communication to the running system which then exposes it to exploitation. Below are the potential attack vectors that a cloud computing systems has [15].

If an attacker can read or modify data that is on a system or moving between systems then the data is vulnerable to deletion or manipulation. Data can be described in two ways, data at rest or data in transit. Data at rest is when data is stored on a system while data in transit is when data is moving between two destinations. If data can be read or altered while in those states by an unauthorized individual then the data is at risk.

Account or service hijacking is when an unauthorized user is able to perform actions as another authenticated user. This may occur from insecurities with accounts or with insecurities revolving around authentication. An attack may be able to impersonate another users activity or manipulate a system to provided sensitive information that is normally not presented to the user.

For systems that are exposed to the Internet, an adversary may try to perform a denial of service attack on the system. This type of attack tries to overwhelm the system and not allow legitimate users to access the systems resource.

Data breaches occur when an unauthorized user is able to access sensitive data that is stored in databases. Databases usually contain the most sensitive information that attackers can then sell. As cloud systems are typically accessible from the Internet, attackers attempt to probe the security of systems to see if they can gain access to the back-end databases.

Attackers also may use cloud systems to send spam or perform computationally intensive tasks (like BitCoin mining) for profit. They access a system via a vulnerability and then have scripts run that perform these tasks automatically.

2.7 SNIC Science Cloud

At Uppsala University there is an OpenStack deployment on the UPPMAX super computer. This is allows professors and students to utilize cloud resource for projects and courses. This is a collaboration project under the Swedish National Infrastructure for Computing (SNIC) [2]. This collaboration is between several institutions to provide large scale cloud computation resources to academia. For the cloud portion of SNIC the regions are located in Goteborg, Umea, and Uppsala. A potential user first requests for a project on the OpenStack system. Once granted they are allocated some amount of resources for users to use on the project. They can then log into any of the regions and spin up their needed environments.

3 Problem Description

In this work, we will be looking at ways to protect VM clusters by ensuring proper configuration steps are set up and used with the addition of looking at ways to implement an intrusion prevention system (IPS) solution into the cloud environment. Security of computer systems is a large field. There are many different areas in the computer field and each one produces individual

[2]http://www.snic.se/

security challenges. For this project, we focused on cloud infrastructure. Computer infrastructure exposed to the Internet face a large number of cyber attacks daily[20]. For private clouds, this is an ongoing problem as many of the users may not know of proper security protocols and leave ports such as secure shell (SSH) open. This in general may not pose a risk, however combined with other misconfigurations vulnerabilities may become present. These types of security vulnerabilities would allow an attacker to gain full control of the virtual machines.

To protect the public cloud, such as OpenStack, the mechanisms are non-existent or weak. The administrators could attempt to monitor network traffic or install specific software on each VM. These solutions are either difficult to set up and maintain for cloud administrators, or only passively work. Scanning network traffic as it goes to and from a computer does not prevent threats but tries to catch ongoing attacks [50]. This may lead to vulnerabilities present in VMs that the administrators do not know about. It is also impractical having to deploy a security solution to every VM that is started on the system. Users may not turn these protective systems on or may require different OS versions that are not compatible with a security system. One of the benefits of the cloud is allowing users to set up and deploy computers with ease, but adding additional factors may discourage users. Therefore the purpose of this project was to create an active system that can monitor other VMs for potential vulnerabilities. Users and administrators do not want to have VMs taken over by malicious users that then perform actions like sending spam or stealing information. The solution presented in this work is created to identify vulnerabilities in VM configurations and alert the administrators before the vulnerability is exploited.

4 Related Work

Before looking at cloud computing and security issues related to the specific field it is a good idea to look at what had been done in the past to secure a standard server. A typical server is placed behind a firewall with an intrusion detection system (IDS) [25], either a network intrusion detection system (NIDS) or a host based intrusion detection systems (HIDS), running within the configuration. These systems are usually only run by administrators who have knowledge of how to maintain large Internet facing systems. One of the main goals in protecting these types of systems is to reduce the ways that an attacker could get into the system. The firewall is used to filter traffic and ensure that only packets of a particular type are passed to the server while other packets were dropped. The IDS system inspects traffic for malicious content and notify administrators if any is found. This model is a fitting standard for how systems facing the Internet should be set up. However, as users typically gain root access to the provisioned VMs it is much harder to have these types of machines sit behind a firewall and IDS. Different users may require different ports open that run different services, therefore a firewall has to be configured for each user. [3] This leads to many VMs being directly exposed to the Internet and attacks. Due to this cloud security had been a hot topic in recent times [52, 36, 32]. These particular papers focus on the different aspects and concerns that are present when running in a cloud environment. They are a appropriate starting point to look at the threat model in the cloud.

In addition an important distinction between a centralized and cloud environment is how information is treated, a good article that describes this is "Addressing cloud computing security issues" [52]. The three pillars of information security are confidentiality, integrity, and availability. In the cloud new difficulties come up for each of these concepts as data in the cloud may move around without the user's knowledge. The following articles are looking at intrusion detection systems (IDS), more specifically how these can be used within the cloud [37, 45]. These articles do not focus so much on implementation as on the theory. An interesting comparison that shows the advantages and disadvantages of different IDS systems are presented table 2 in "A survey of intrusion detection techniques in Cloud" [37]. This is the basis for the types of IDS solutions that were chosen for this project. As this work was primarily focused on OpenStack, one particular IDS conference talk was used as a starting point for this research [33]. The talk is about implementing a type of NIDS into the OpenStack ecosystem and some of the challenges faced in creating such a system. The system tries to replicate a typical NIDS however using virtual networking and other OpenStack specific components. One particular challenge that stood out was how the NIDS needed to be able to cope with much more varying traffic scale. As the cloud is dynamic, the NIDS has to be able to scale as well.

[3]Users can elect to create their own virtual firewall in the cloud. The users then can set up specific rules to restrict different type of traffic similar to how a physical firewall would work. This would however need to be set up in a project and the VMs would need to be placed behind it.

The type of network and system monitoring that is proposed in this work is not new. At Cambridge University a set of probes are running on networks to ensure no security holes and ensure administrators know what is running on the network [8]. The intrusion prevention systems, at Cambridge, will run and scan the other networks to find any potential weaknesses in the systems. If a vulnerability is found an alert is generated and sent to an administrator. This allows for a vulnerability to be found before it is expected by an attacker. The work done during this project follows the same principles but is adapted to the OpenStack cloud environment. The Cambridge system worked with a normal computer and network system, while the one created in this project focuses on probing machines that are created in an OpenStack project. The created system is therefore able to communicate with the OpenStack layer via the backend API. This allows the system to pull information about machines and some information about their configuration. This allows for the administrator to query the system and have the information about the threat alongside information about the running information. This can tell them who the machine belongs to, what version of the OS is running, up time, etc. without having to search for this information in several places or programs.

There are several commercial and free solutions that can be set up to perform similar types of actions as well. Two such tools are Nessus [9] and OpenVAS [11]. These tools can be set up on a network so that they scan and monitor the network for known vulnerabilities. Nessus and OpenVAS have similarities to what Cambridge University uses, probes are used to identify potential vulnerabilities in deployed systems. However these systems were not designed with integration to an OpenStack system or use in a cloud environment. In this research, a similar type of probing was created to work with the OpenStack cloud platform.

5 OpenStack

OpenStack is an open source platform for cloud computing [7]. OpenStack is built of many components that are designed to provide a set of services (Nova, Neutron, Swift, etc.), these are described below. The full list can be found on the OpenStack website [4]. OpenStack is very scalable and diverse system that can be arranged to fit the needs of any cloud environment. For this

[4] https://www.openstack.org/software/project-navigator/

project we ran OpenStack newton version 15.0.0.0.rc1. OpenStack is very modular with several core features running as the backbone of the software. The features that were used for this project were: Nova, Neutron, and Swift.

Nova is the service that provides access to OpenStack compute resources. It is the base service that a user would interact with to run, configure, and destroy virtual machines. The compute plan manages how resources are distributed to different projects that run on the OpenStack. Recourses are allocated by Nova to different VMs.

Neutron is the service that controls and configures all of the networking between machines and the external network. This interlinks the Nova instances with the physical networking infrastructure. Neutron is also used to allow the other services to talk between one another. Therefore the Nova service can communicate with the Swift storage service via the Neutron networking service.

Swift is the platform that is used for object and blob storage. It provides a large space for images, snapshots, and other storage needs to be assigned to. Object storage is for adding large static data sets to be requested by from multiple VMS. Blob storage provides a type of storage for information that typically does not conform to a set type of data. Some examples of blob data are video, audio, or images.

One of the main aspects of OpenStack is a tool called Heat [10]. Heat is a orchestration tool that sets up an environments though a script file. Heat will take is a file that has commands on how a environment should be set up. The file will contain instructions on how machines, networks, storage, etc. should be set up and configured together. As the script file is read, resources such as servers, floating IP addresses, and security groups are created. This tool allows complicated environments to be easily set up and deployed by the administrators. On the OpenStack website a community has been created to publish these helpful scripts [5]. This causes a problem as people use the scripts that are their posted without full knowledge of what is happening in the environment. This may lead to services and machines exposed to the Internet without the user knowing about them.

[5]https://github.com/openstack/heat-templates/tree/master/hot

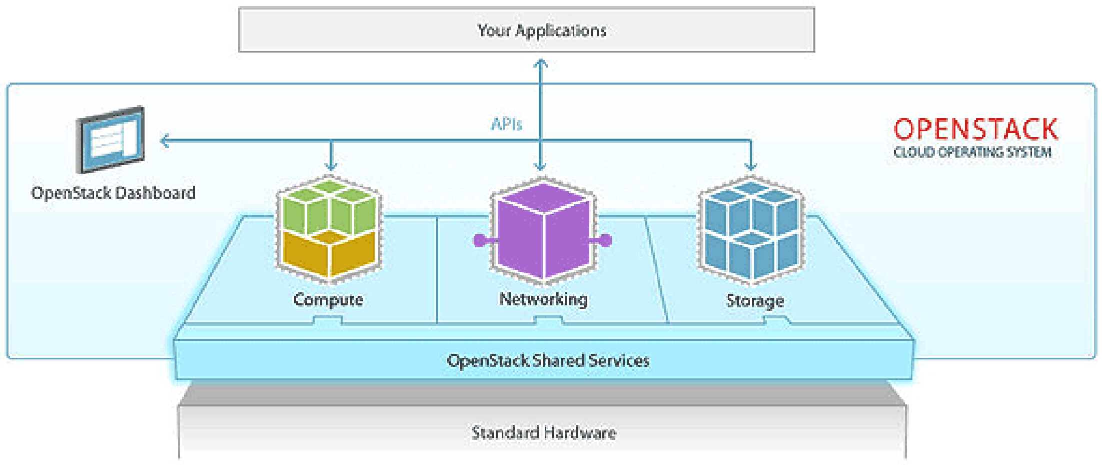


Figure 4: OpenStack Cloud System

6

6 Security

The general focus of computer security is to ensure that computer systems follow the CIA model of confidentiality, integrity, and availability [32]. "The design artifacts that describe how the security controls (security countermeasures) are positioned, and how they relate to the overall IT Architecture. These controls serve the purpose to maintain the system's quality attributes, among them confidentiality, integrity, availability, accountability and assurance." [32]. The disciplines that are focused on in this paper include: intrusion detection systems, network security, and system security. Intrusion detection systems come in two main forms, host based and network based. A host based system runs on the host and attempts to detect any security threats on the host machine. A network based IDS is connected on the network and inspects network traffic, trying to find threats by inspecting network traffic patterns. Both of these systems have advantages and disadvantages. With a host based system software must be installed and configured on each system. In a network IDS system the packets are monitored for unusual traf-

[6]https://dzone.com

fic patterns. There are disadvantages to this type of solution also, mainly network traffic can be encrypted and the overall volume of traffic encountered. Some examples of network based IDS tools are Snort. A large portion of the problem for computer security comes from the networks that computers are attached to. The network allows other computers to communicate and attempt to access the particular machine. This is greatly increased if the computer is not placed behind a firewall/rougher and is directly accessible form the Internet. If an Internet facing computer is not updated properly vulnerabilities can be used to run unintended programs. A unintended program running on a systems can be a concern. These programs can be used to send email spam, steal credentials, or participate in larger network attacks.

6.1 Network Security

Network security aims to ensure that the underlying communication between hosts is secure. That may be to ensure that messages passed through the network are untampered or not intercepted, ensure a device is properly protected from attack, or to divert/stop an attach reaching a machine. As this is a large field in itself we will break it down into smaller sections. Computers need to communicate with each other to relay information, this is done through the networking layers [19]. An attacker could, using different methods, intercept traffic and/or manipulate it. Network security in this context relates to ensure that information is preserved while in transit. For cloud security this type of network security is important to ensure that communication from a user to the machine is kept secure. Network security is also meant to protect machines from external network threats. This usually is done through a network firewall, which will sit between a external and internal network allowing only certain trusted traffic through. As most cloud machines may be exposed to the Internet, ensuring that traffic to the machines is trusted can be difficult. Most cloud based machines are operated through SSH and may need other ports open to communicate with other services or machines. This creates a network security problem as external open ports will be repeatedly attacked by botnets [22]. The last layer of security that the network can provide is protection against flooding or distributed denial of service attacks (DDoS attacks). These attacks try and overwhelm a server and take the service offline.

6.2 System Level Security

System security aims to protect running systems from malicious code execution. A malicious program is typically comprised of 2 pieces, an exploit and a payload. An exploit is a small mechanism designed to gain access to the system. Once access is gained the payload will be retrieved and executed. The payload then will perform actions like, lock file, upload user data, or capture keystrokes. Protection against this type of attack is mainly done through a host based intrusion detection system (HIDS) these are typically known as anti-virus or anti-malware. A HIDS is a small program that runs on a system scanning memory, disks, running processes, etc. If one of those matches a predefined rule then that code is quarantined and removed. These type of rule based systems lack in one particular area, they are bad at detecting new threats until a rule is created. Therefore newer HIDS try and use some type of machine learning or prediction to assert whether malicious code is being executed. Typically a security system will be set up on individual computers or servers. It will then run in the background protecting the system. In cloud security HIDS are a problem [38] as a wide variety of VMs may be created with users not knowing how to properly set up systems. Users may not know how to set up HIDS systems on these machines or the OS versions may not even support a HIDS.

6.3 User Level Security

User level security relies on how well users understand threats and access risks [48]. As users are the main proprietor of systems they hold the keys to ensuring that a system is safe from malicious adversaries. It however is problematic as many users do not understand system security and are not interested in learning it. The users then perform actions that compromise system security such as using weak passwords, opening ports to the Internet, and running unknown programs.

6.4 Cloud Security Risks

Cloud security posses several risk that are different from what a typical server might encounter. As all of the cloud systems are orchestrated and deployed via a web page there is a possibility of account takeover. Account takeover occurs when an attacker is able to gain access to the users profile and perform action's on the users part. An example of this could be if a user uses a weak password to protect their account. This would allow an adversary to deploy machines while adding the attackers own private keys which would allow them to access the machines. In addition a user may not protect their private key in correct manner. If an attacker was able to gain access or steal the private key, they would be able to log into any of the users machines. They would then be able to run their own programs or steal any data that is on the server. Deployed machines are usually orchestrated and created by users and not teams of administrators. The user may not have the knowledge in secure computing, thus the system may be deployed in an insecure manner.

7 User Recommendation

The first stage of this project involved setting up user recommendations for configuring a secure VM and understanding some security features in the OpenStack platform. This involved taking the two main problems that the SNIC team faced: poorly secured SSH access and open ports. The recommendation describes what SSH is and how to use proper key pair authentication. The guild then goes onto describing ports and how to configure good security rules in OpenStack to prevent exposed ports. Below is a shortened list while the full readme can be found in the appendix or following link [7].

- SSH is the mechanism in which one should connect to cloud systems. It is a set of tools and protocols set to help ensure a secure connection between one's computer and server. It is one of the standard protocols to security communicate with a remote computer. The tool creates a secure link between the two machines, allowing for encrypted communication to occur.

- A proper SSH key pair should be used. The key should be at least 2048 bits long and should have a password on it. This will ensure that even if the key is compromised no one would be able to use the key to get into any systems. One should never use user name and password authentication over SSH. There are a lot of bots on the Internet that attempt to break into systems that have weak user name and password combinations in SSH.

[7]https://github.com/DarkLog1x/MasterThesis/tree/master/UserRecomendation

- Once a key pair is created and the key is uploaded the private key should be securely held on the users computer. The user should ensure that the private key only have read and write access by root. This protects the key even if the users account was compromised. The user should also never post or send the private key to any one. The key should be held only on the users computer.

- On the OpenStack site, users should use correct security groups for their machines. A security group is a way to define what ports a VM can access or be accessed by. This helps prevent unwanted exposure to ports on the machines. A correct security group will consist of only the used ports allowed to be open while the rest would be closed. This would prevent ports being accidentally opened.

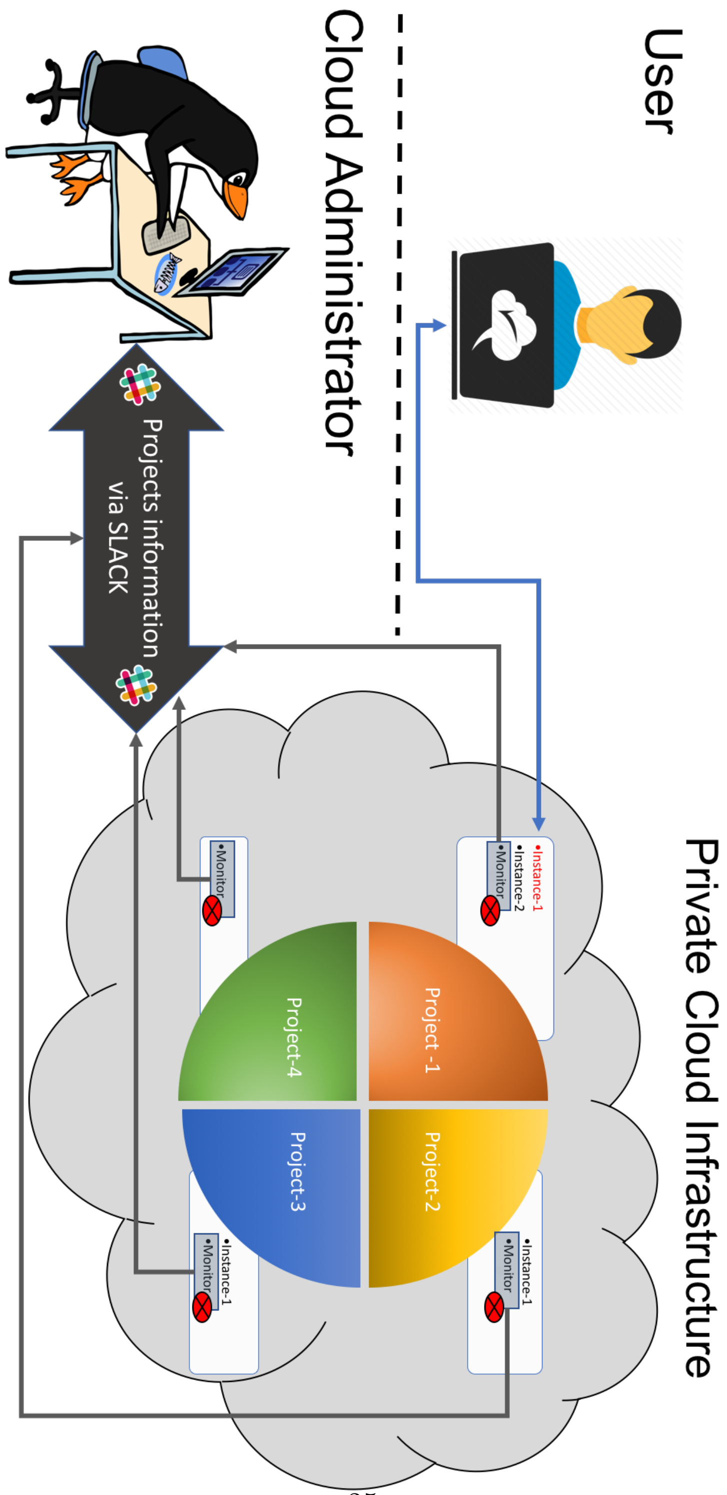
- One should be aware of what services are run on each VM. These services may have open ports that increase the attack surface of the VM. These should either be turned off or the port should be blocked or filtered using the local firewall on the computer.

- Users of VMs should have a security mindset. This will greatly lower any potential vulnerabilities and is one of the best ways to prevent system compromise. Users should be aware of the tools that they use, programs they configure, and run on the systems. Users should not copy commands from the Internet and simply run them but should understand what the command does.

8 Design Implementation

The next part of the project focused on creating a system that would allow for ensuring that the user recommendation was followed. A key part of this project was to create a simplistic interface that would allow an administrator to view changes to the environment and query for more information. Some base goals were created: scalability, ease of use, and intuitiveness. As the system is designed to be deployed over several cloud groups we needed a messaging system that could handle such a task. Email was first considered however it was soon realized that email would not work well due to the frequency of data sent and then inability to interact with the service. The next consideration was Internet Relay Chat (IRC) [29], however as IRC is relatively unused in the professional work environment and users would need to learn how to interact with the protocol that idea was abandoned. Therefore as a final solution it was decided to work to build the messaging solution using Slack [47]. This matched all of our starting criteria and allowed for a clean and simple implementation. Slack is a messaging application that allows groups of users to interact with one another in channels or directly. One of the really neat features of this is the ability to program bots for users to interact with. Therefore the decision was made to create a bot that would interact with a channel for each cluster of machines. The bot would update the channel with any new information that the scan found ensuring that a constant flow was preserved. Users are also able to query to bot directly and get more information about the environment. This allows seamless integration between getting information from the environment and being able to dive deeper and investigate a machine.

For the first part of the project a modular design was created to enable further types of scanning to be added. An important aspect when designing the "watchdog" system with as little privileges as possible to prevent any unnecessary security holes. The watchdog is a phrase that was used throughout the project to describe the machine that performed all of the scanning. The following is a diagram of an example cloud environment.


User
Cloud Administrator
Private Cloud Infrastructure
Projects information via SLACK
•Instance-1
•Instance-2
•Monitor
Project -1
Project-2
•Instance-1
•Monitor
Project-3
•Instance-1
•Monitor
Project-4
•Monitor

The ”watchdog” is initiated in the cloud cluster with a configuration that the administrator creates. This configuration is described in section 9 ”Architecture” of the report. When the VM is initiated it will connect to the OpenStack Control Plane and get the list of servers, IPs, and floating IPs. Once that information is retreated the different modules will be run. These modules are the different scanning tools that are used to detect improper configurations. In the project two modules were created that used the nmap and ssh_scan programs to find vulnerabilities. Nmap is a tool used to scan a local or external network and view what machines are running on it. Nmap will be able to scan the machines ports and tell which ones are open and what services are running on them. It will also try and guess heuristically the operating system that is being run on the machine. This tool is therefore used in this system to find machines that have open ports and to see what services are run on them. The ssh_scan tool is used to ensure that a public/private key pair is used and no other authentication method is present. It will scan port 22 and alert if any other form of authentication is used. This data from the scans is saved into a database that can be later queried by the analysis and reporting portion’s of the system. Once the data is gathered it is compared to the configuration file. If a discrepancy is found between the expected results and the scanned machine an alert is sent to the administrator. For this project the administrator receives notification on a separate slack channel. Slack is described in more detail in the Tools section of this report. This entire process is run every 20 minutes by a cron job located on the watchdog machine. This type of design allows for a flexible platform that can have each component further customized. The code for this part of the project can be found at https://github.com/DarkLog1x/Groot

9 Architecture

For the monitoring resource an important architecture design was to ensure that the system was modular. This enabled the system to be adapted to different cloud environments. Users would be able to create individual monitoring tools or logging techniques to be used for configuring to specific infrastructures. As for the proof of concept project Nmap and ssh_scan were implemented.

The main portion of the entire program is the database. As the system is designed in a modular manner all the data is stored and retrieved from the database. The database is a MongoDB [39] instance that has one collection that hold all the data from different scans. The only data that is needed inside of the database is the server ID. This is the key to which all other data about a particular machine is added to. When data is stored the ID of the VM is passed along with the data. When a user requests data they must also provides the ID for the data they would like.

Apart from the database the other modules are started from the main python program. To do this, the system will first connect to the OpenStack backend via credentials provided in a configuration file. This will pull down all of the VM IDs. This is then fed to each scanning model. These modules intern can query the backend for any additional information that they need. To illustrate this the Nmap modules will use the provided IDs to obtain the internal and external IP address for each machine. This will then feed the query to the Nmap program. The Nmap model will then obtain the data and parse out the needed information, such as open ports. Once the data from the model is as wanted it will be sent to the database to be added for each VM.

To add a new model the main function must be edited so that it will call the new model. The output from the new model must follow a set standardized of $(ID : "IDnumber", data)$ this is then sent via the main function to the database model to be inserted into the database.

A configuration file is used to tell the program how a particular system in the project should look like. An administrator must first create this file for each project. This configuration file is set for the system to know what type of system configuration the servers should have. Below is a typical configuration:

```
# "*" special case to print all other open port
# The "#" represents commented lines
ports: 22 - open - 7.2p2 Ubuntu 4ubuntu2.1
ports: 80 - closed
IP: ip_external_auth - publickey
IP: ip_internal_auth - publickey
```

The file is read in line by line. The system will then parse each line, splitting each line via the : and $-$ character. The first part before the :, is matched with the different scanning modules. In the examples case this will match Nmap for "ports" and ssh_scan for "IP". When the modules were

Table 1: Configuration file breakdown

Module specification	Relevant information	Expected results	Extra options
ports	22	open	7.2p2 Ubuntu 4ubuntu2.1
ports	80	closed	None
IP	ip_external_auth	publickey	None
IP	ip_internal_auth	publickey	None

designed these were chosen as the trigger words for each of the module, if wanted by a user they could edit the modules to change this. Thus allowing each one to only recognize the lines of the configuration meant for its task. The part after the : represents what the scan will be looking for. In the case of the first test, this will be port 22. The next command looks at what the expected result should be. In the example above that would mean that the expected result for port 22 would be that it "open". If the result of the module scan does not match the expected output, a message is created and sent to the communication mechanism, in this case Slack. In the example above port 22 is expected open with the version of "7.2p2 Ubuntu 4ubuntu2.1". The version information is additional information that the Nmap scan can take in and use. The information provided in the configurations needs to match the output of the of the used scans. This is due to the fact that the content's of the configurations file are string compared to the outputs that the modules produce.

There are two spacial characters in the configuration file that can be used. The first one is the "#" symbol, this is used at the start of the configuration file to mean that the line is a comment. The second character is the "*" which denotes all of the ports in the Nmap module. That is to say a rule can be written like the following: "ports: * - closed". This would mean that all of the ports are expected to be closed, and if any ports are open the system should create a message to notify and administrator.

The Slack bot uses the slack application programing interface (API) to communicate with the Slack web interface. The API allows for messages to be sent between the bot and the Slack servers.

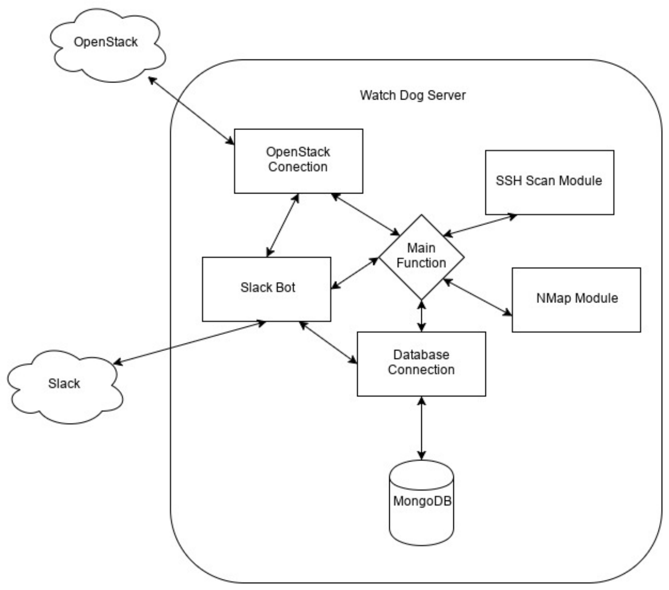


Figure 5: Intrusion detection system architecture, the main component is the "Main Function" and the database. All information is fed through the main function allowing features to easily be interchanged and expanded.

9.1 Tools

For this project several external tools that were used. These tools were picked to perform specific tasks like collect SSH configuration data.

9.1.1 Nmap - Network Scanning Tool

Nmap is a network scanning solution that can be used to scan for open ports, running services, network topography, and others [8]. Nmap allows for scanned data to be saved into an ẋml file that can then be filtered and input into the database.

9.1.2 ssh_scan - SSH Security Auditing Tool

Ssh_scan is another tool used to gather information about remote SSH configurations. It is able to detect the type of authentication, SSH protocol used, and several others [9]. The output is saved to a json file that is then read into the database.

9.1.3 Slack Bot

Slack bot [10] is a small bot built on top of the Slack API to display information form the database and query data in real time. The bot will connect to a channel specifically created for each cloud cluster. There the bot will print relevant information from the database. One of the advantages that Slack provides is the ability to communicate between the users and the bots. Users are also able to query the database via the bot, this allows an administrator to quickly find a problem and find information about that machine with ease.

9.1.4 Ansible - Server Orchestration Tool

Ansible [11] is a tool designed to automate system deployment. The tool has a scripting language that describes steps for systems to be configured.

[8]https://nmap.org/
[9]https://github.com/mozilla/ssh_scan
[10]https://api.slack.com/bot-users
[11]https://www.ansible.com/

```
---
- hosts: all
  remote_user: ubuntu
  gather_facts: no
  pre_tasks:
    - name: Update and install python
      with_items:
        - python
        - python-dev
        - python-setuptools
        - git-core
        - gcc
        - nmap
        - ruby
        - ruby-dev
        - libsqlite3-dev
        - make
        - htop
        - nload
        - mongodb-org
        - sqlite3
```

In the above example for all hosts in a separate host file the system will login as "ubuntu" and install the list of programs using `apt-get`. The default connection ot each host is done via SSH where the user points the ansible program to their private key. This tool is used to deploy complex environments with little hassle. As this project was also designed to be easily deployed ansible was chosen. There are several other tools such as Puppet [12] or Chef [13]. However prior knowledge and usability determined the software choice.

9.1.5 Git - Source Control

An important tool used throughout this project was Git. Git is a version control system that keeps track of all changes to files. In addition it allows

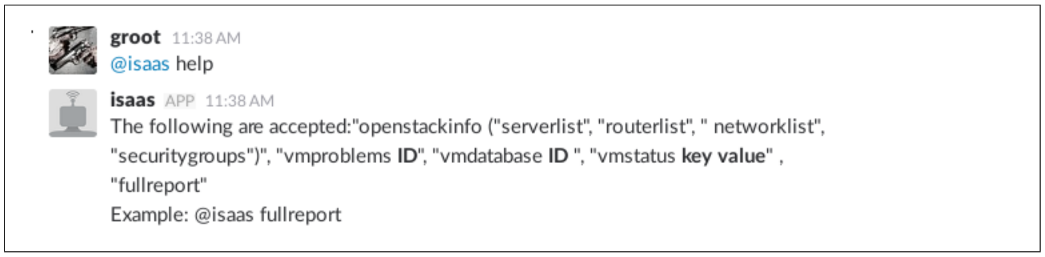
[12]https://puppet.com/
[13]https://www.chef.io/

for uploading of these changes to websites like GitHub[14] and Bitbucket [15]. This creates a viewable time line of all changes to the code base as well as provides a backup in the cloud in case of a system failure.

10 System Interaction

One of the important aspects of the system that particularly had focus was interactability. We wanted to present the administrator with a tool that would allow them to gain all the information that they wanted about a system. We wanted the user to be able to interact with a bot, and have the bot respond with the relevant information that the user wanted. The system should be fluid and informative, without the user having to switch between different programs to find relevant information. To communicate with the bot, the user simply calls the bot using the @ sign and then the command.

The picture below show the help command.

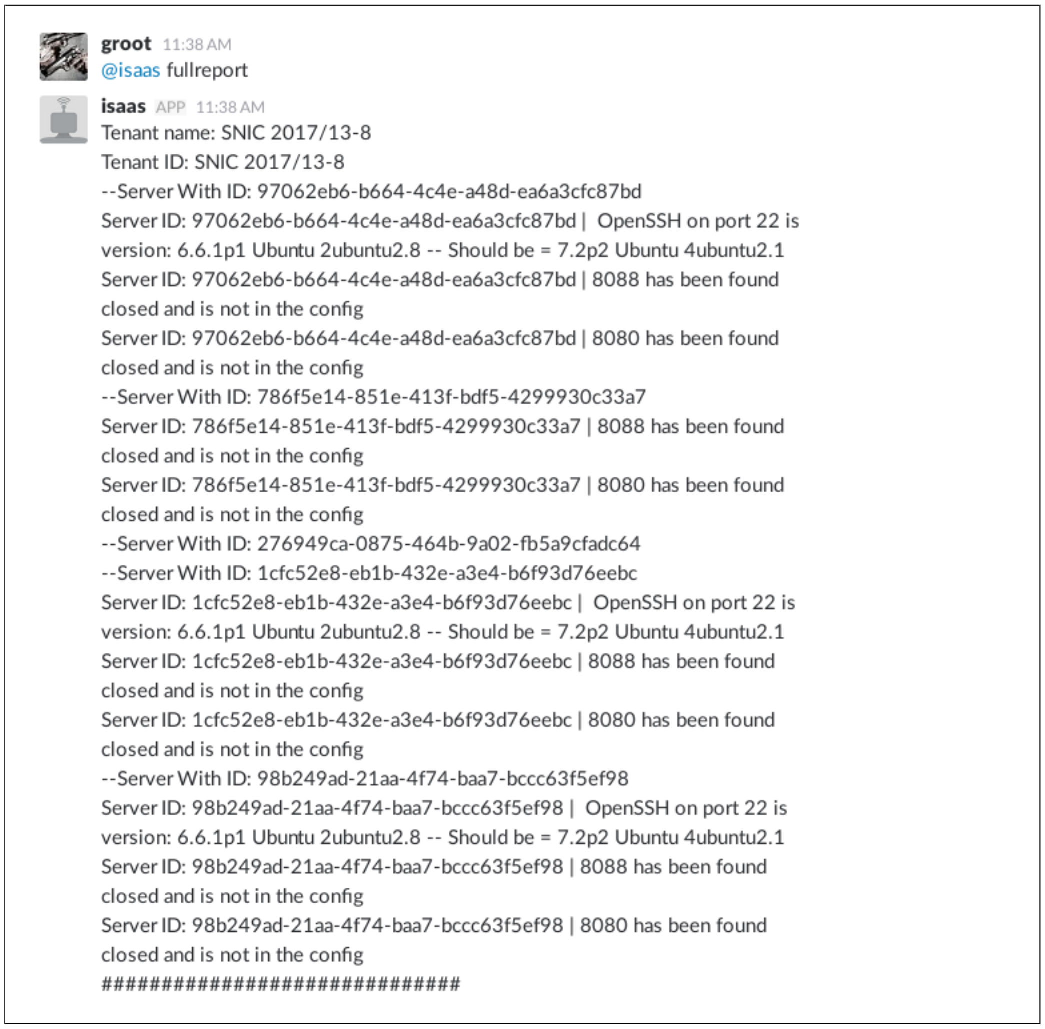


groot 11:38 AM
@isaas help

isaas APP 11:38 AM
The following are accepted:"openstackinfo ("serverlist", "routerlist", " networklist", "securitygroups")", "vmproblems **ID**", "vmdatabase **ID** ", "vmstatus **key value**" , "fullreport"
Example: @isaas fullreport

[14]https://www.github.com
[15]https://www.bitbucket.org

The picture below shows how a full report for a project can be generated.

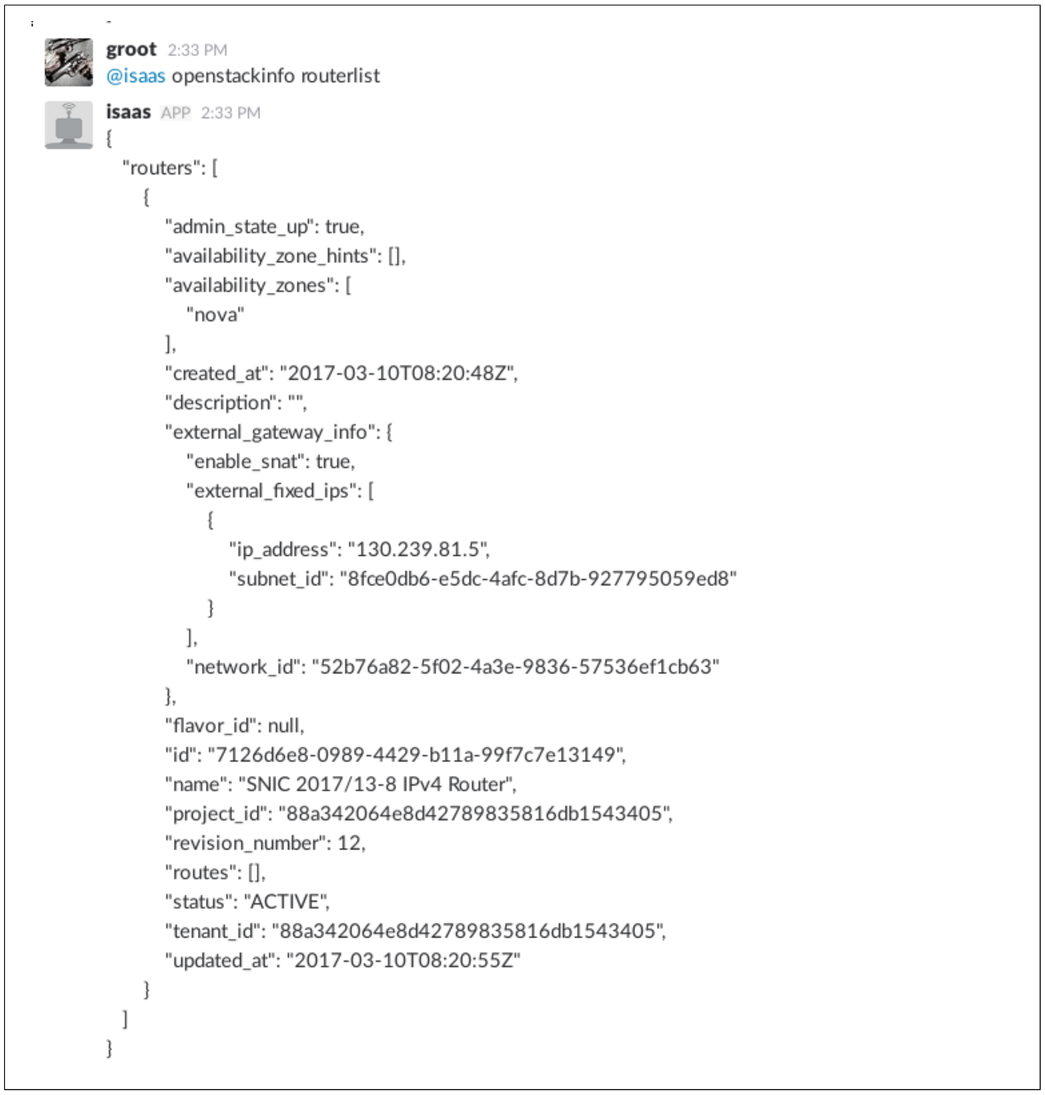


groot 11:38 AM
@isaas fullreport

isaas APP 11:38 AM

```
Tenant name: SNIC 2017/13-8
Tenant ID: SNIC 2017/13-8
--Server With ID: 97062eb6-b664-4c4e-a48d-ea6a3cfc87bd
Server ID: 97062eb6-b664-4c4e-a48d-ea6a3cfc87bd |  OpenSSH on port 22 is
version: 6.6.1p1 Ubuntu 2ubuntu2.8 -- Should be = 7.2p2 Ubuntu 4ubuntu2.1
Server ID: 97062eb6-b664-4c4e-a48d-ea6a3cfc87bd | 8088 has been found
closed and is not in the config
Server ID: 97062eb6-b664-4c4e-a48d-ea6a3cfc87bd | 8080 has been found
closed and is not in the config
--Server With ID: 786f5e14-851e-413f-bdf5-4299930c33a7
Server ID: 786f5e14-851e-413f-bdf5-4299930c33a7 | 8088 has been found
closed and is not in the config
Server ID: 786f5e14-851e-413f-bdf5-4299930c33a7 | 8080 has been found
closed and is not in the config
--Server With ID: 276949ca-0875-464b-9a02-fb5a9cfadc64
--Server With ID: 1cfc52e8-eb1b-432e-a3e4-b6f93d76eebc
Server ID: 1cfc52e8-eb1b-432e-a3e4-b6f93d76eebc |  OpenSSH on port 22 is
version: 6.6.1p1 Ubuntu 2ubuntu2.8 -- Should be = 7.2p2 Ubuntu 4ubuntu2.1
Server ID: 1cfc52e8-eb1b-432e-a3e4-b6f93d76eebc | 8088 has been found
closed and is not in the config
Server ID: 1cfc52e8-eb1b-432e-a3e4-b6f93d76eebc | 8080 has been found
closed and is not in the config
--Server With ID: 98b249ad-21aa-4f74-baa7-bccc63f5ef98
Server ID: 98b249ad-21aa-4f74-baa7-bccc63f5ef98 |  OpenSSH on port 22 is
version: 6.6.1p1 Ubuntu 2ubuntu2.8 -- Should be = 7.2p2 Ubuntu 4ubuntu2.1
Server ID: 98b249ad-21aa-4f74-baa7-bccc63f5ef98 | 8088 has been found
closed and is not in the config
Server ID: 98b249ad-21aa-4f74-baa7-bccc63f5ef98 | 8080 has been found
closed and is not in the config
##############################
```

The picture below shows how to call another bot and retrieve a projects' router information.



groot 2:33 PM
@isaas openstackinfo routerlist

isaas APP 2:33 PM

```
{
  "routers": [
    {
      "admin_state_up": true,
      "availability_zone_hints": [],
      "availability_zones": [
        "nova"
      ],
      "created_at": "2017-03-10T08:20:48Z",
      "description": "",
      "external_gateway_info": {
        "enable_snat": true,
        "external_fixed_ips": [
          {
            "ip_address": "130.239.81.5",
            "subnet_id": "8fce0db6-e5dc-4afc-8d7b-927795059ed8"
          }
        ],
        "network_id": "52b76a82-5f02-4a3e-9836-57536ef1cb63"
      },
      "flavor_id": null,
      "id": "7126d6e8-0989-4429-b11a-99f7c7e13149",
      "name": "SNIC 2017/13-8 IPv4 Router",
      "project_id": "88a342064e8d42789835816db1543405",
      "revision_number": 12,
      "routes": [],
      "status": "ACTIVE",
      "tenant_id": "88a342064e8d42789835816db1543405",
      "updated_at": "2017-03-10T08:20:55Z"
    }
  ]
}
```

11 Tests

To ensure that the system functioned as intended, a two fold testing scenario was implemented. The first phase of testing was done in a controlled environment. The second phase was having the created system deployed in a production environment. The tests were designed in two ways, first to ensure that the created system was working as desired and second that it was able to work in a production environment.

The first test was used as a way to ensure that the created components of the system were working as desired. For this to be achieved a set of test goals was first created.

- To recognize when a new VM was created.
- To scan the VMs with different probes (i.e. ssh scan and nmap port scan).
- To synthesize the gathered information and compare it with the desired VM configuration.
- To send the data to the administrators via Slack.

The "watchdog" system was then designed and coded to meet these set goals. For the first test, a separate OpenStack environment was created, this ensured that the environment could be controlled and only specific VMs would be turned on and running at a time. The test consisted of creating a "watchdog" VM (running on Ubuntu 16.04) on which the created software would be run. The "watchdog" VM was set up by running the Ansible configuration script. The configuration for the "watchdog" would be configured to accept Ubuntu 16.04 VM images but would raise alerts once other types of VMs were run. The Slack service would be run and an administrator was added to the group. Then several other VMs were initialized in the project environment. The "watchdog" VM was then spun up and the code was deployed. New VMs of different Ubuntu versions where spun up and then destroyed with some VMs having new services such as MySQL started and exposed. The running "Watchdog" system would then pick up on these changes are report them to the administrator.

The second test was used to see how the system scaled with more users and how it handled in a non test environment. The test was done in an project were approximately 50 students were taking a big data course. The students had labs that used VMs were specific ports and services would be run. In this project the "watchdog" service was deployed. As the service is designed to be passive and not intrusive there was no possibility of affecting running systems. The system was able to detect several instances where students deployed systems incorrectly which could potentially lead to compromised VMs. This showed that the system scaled to meet larger demands and that it was able to detect miss-configured VMs in a real world scenario. This test proved to be able to find student systems that had missconfigurations in them. The results of these tests are described in more detail in the next section.

12 Results

As the project consisted of three parts each part is separately evaluated in this section. The first part, user recommendation, describes the compilation of different resources into a final user guide. The second section describes the results for the created IPS system in a real world use case. The third part talks about how the imaging tool works and presents a use case for administrators.

12.1 User Recommendation

A user recommendation was written to help users properly connect to initialized VMs. The recommendation focuses on how to connect to the VMs using a proper SSH key. The guild first walks through how to set up a SSH key pair on a users local machine. This ensures that a user creates a proper strong key pair. The user is then instructed on how to add the public key to the OpenStack interface. Along with the SSH connection other simple recommendations are provided to the user. These are to help a user understand the different security implications no matter what they use the VM for. Some examples of what the recommendations looks at are, open ports and security rules. The user recommendation can be found at the following url [16]. The recommendations come from several years of security work in the field and an accumulation of other sources (outdated however still has good general advice [18]) and [26]. As of the start of 2018 these are considered the first steps in securing a server that is exposed to the Internet.

[16]https://github.com/DarkLog1x/MasterThesis/tree/master/UserRecomendation

12.2 Intrusion Prevention System

The overall results for the experiments turned out well. The system was able to be easily set up and run, this allowed the administrators to have one place to view the status for a given project. If an event occurred that needed to be taken look at, the needed information could easily be queried. The system did manage to catch systems that were deployed in a production environment that did not match the predefined configuration file.

12.2.1 Intrusion Prevention System Results

For both tests the same configuration for the scanning was used. This was done to ensure that changing of the configuration did not affect the outcome of either test. The image below shows the configuration for the modules used in both the controlled and real world test.

```
#This is a config file
#It needs to be filled out similer to the output of the script
   that is run (a.k.a - NMAP, ssh\_scan, etc)
#* special case to print all other open port
ports: 22 - open
ports: 80 - closed
ports: 443 - closed
ports: 5000 - closed
ports: 6000 - closed
IP: ip_external_auth - publickey
IP: ip_internal_auth - publickey
```

The ports 22, 80, 443, 5000, and 6000 had values in which the expected configuration of the server matched with the created configuration file. If any other ports were opened by a service or one of the later ports was not in the correct state. A message would be created and sent to an administrator via Slack.

As the first test environment was used for design and development with very controlled machine states. It was decided to show some of the results from the second phase of testing. The second real world test was the main test environment were the following results were gathered. Once the system was set up in the project and verified that things were working as they should be, the system was left to gather and display information about the state of machines in the project. After running for a while inconsistencies in the VM configuration started to be found. A small snippet of the Slack channel is shown below as there was a lot of data sent. In this section 5 machines are shown, 3 of them have inconsistencies that are not in the configuration file.

The table on the next page describes how the output of the Slack bot will look like. Below the table is the output from the Slack channel.

Table 2: Slack Ouput Breakdown

Line Number	VM ID	Description
1.	930ee1bb-e07a-461f-b09f-1e1a9c864a36	There are 3 open ports (lines 2-4) that should be closed.
5.	f57b9369-4e15-4151-9b0b-7d405cbdb5e4	The VM allows for user authentication via a username and password.
10.	81933a08-9451-4d16-aca4-9fb6269b4e0d	There are 3 open ports (lines 11-13) that should be closed.
9. and 14.	ad74be2b-e452-43df-b307-d8ad3bcd6cb7 and 6e19b2a1-1dbd-4fe1-83d6-8e57cdd08ff7	The VMs are running and follow the configuration.

```
1.  --Server With ID: 930ee1bb-e07a-461f-b09f-1e1a9c864a36
2.  Server ID: 930ee1bb-e07a-461f-b09f-1e1a9c864a36 | 8088 has
    been found open and is not in the config
3.  Server ID: 930ee1bb-e07a-461f-b09f-1e1a9c864a36 | 8042 has
    been found open and is not in the config
4.  Server ID: 930ee1bb-e07a-461f-b09f-1e1a9c864a36 | 8031 has
    been found open and is not in the config
5.  --Server With ID: f57b9369-4e15-4151-9b0b-7d405cbdb5e4
6.  Server ID: f57b9369-4e15-4151-9b0b-7d405cbdb5e4 |
   ip_external_auth: publickey password keyboard-interactive
   -- Should be = publickey
7.  Server ID: f57b9369-4e15-4151-9b0b-7d405cbdb5e4 |
   ip_internal_auth: publickey password keyboard-interactive
   -- Should be = publickey
8.  Server ID: f57b9369-4e15-4151-9b0b-7d405cbdb5e4 | OpenSSH
   on port 22 is version: 7.4 -- Should be = 7.2p2 Ubuntu 4
   ubuntu2.1
9.  --Server With ID: ad74be2b-e452-43df-b307-d8ad3bcd6cb7
10.  --Server With ID: 81933a08-9451-4d16-aca4-9fb6269b4e0d
11.  Server ID: 81933a08-9451-4d16-aca4-9fb6269b4e0d | 8088
   has been found open and is not in the config
12.  Server ID: 81933a08-9451-4d16-aca4-9fb6269b4e0d | 8042
   has been found open and is not in the config
13.  Server ID: 81933a08-9451-4d16-aca4-9fb6269b4e0d | 8031
   has been found open and is not in the config
14.  --Server With ID: 6e19b2a1-1dbd-4fe1-83d6-8e57cdd08ff7
```

Once a potential problem was found the *bot* can be queried to get a full report of the virtual machine in question. This can be seen in the output below.

```
Tenant name: SNIC 2017/13-8

Tenant ID: SNIC 2017/13-8

[
{
   "ID": "f57b9369-4e15-4151-9b0b-7d405cbdb5e4",
   "IP": {
      "ip_external": "130.239.81.28",
      "ip_external_auth": "publickey password keyboard-
         interactive",
      "ip_internal": "192.168.1.9",
      "ip_internal_auth": "publickey password keyboard-
         interactive"
   },
   "OpenStack_info": {
      "created_at": "2017-03-16T08:09:08Z",
      "image_id": "e4752852-c053-4fee-b198-6d990d914e3a",
      "image_name": "CoreOS - latest",
      "updated_at": "2017-03-16T08:09:25Z",
      "user_id": "d152c3dbd72c44af95ac4e6e48ccb0ec"
   },
   "_id": {
      "\$oid": "58db779eb39aa86565e2f1f3"
   },
   "ports": {
      "22": "open"
   },
   "services": {
      "OpenSSH on port 22 is version": "7.4"
   }
}
]
```

The administrator then gets a full view of the machine in question and can then decide what the best possible next action is. As mentioned before the type of data that is present is not fixed therefore each module would add its own collected data for each specific VM. Therefore only the relevant data for the machine is displayed. This type of condensed information was not present before this system was developed. Administrators would need to run several programs to gather the data and then would need to synthesizes it by hand.

12.3 Hypervisor Information Gathering Results

Once a compromised machine has been found a copy of the memory, disk, and system information is made. This is to help if the administrators would like to further understand what happened. To do this a script was written that uses the `virsh` tool to connect to the kernel based virtual machine (kvm). Once connected the script pulls relevant data and stores them into temporary files.

The data that the script pulled is a disk image, the memory of the machine, and the cpu state. The disk image saves the state of the hard drive and allows for the machine to be recreated if need be. The memory is saved as a raw memory dump. This creates a large file that represents the active memory from the virtual machine. Lastly the CPU state is saved to show the CPU usage. These 3 items provide a comprehensive overview of what the machine was doing at the time. This would then allow an administrator to further figure out the cause of the exploit. The administrator could use tools like volatility framework [12] to then import the memory dump and then attempt to find what programs and services were running on the machine.

12.4 Overall Expectations

As seen above the created system was able to work as expected. The recommendations allow for individuals to have a quick overview of the best practices in working with VMs while the groot tool allows for ensuring that users follow the created guild lines. When a problem arises with a configuration an administrator can investigate incidents and pull data about a VM in Slack without having to switch over to another tool. This allows for the administrator to have a understanding about what is occurring in a particular project, therefore meeting the expected outcome of the project.

13 Discussion

Throughout the project every attempt was made to ensure that the good design decisions were adhered to. This meant that the system was reliable, fast, and stable. However due to time constraints the code portion of the project will be branded as a prototype. This is because there are several drawbacks that could not have been overcome in this specific time frame.

The current implementation utilizes a individual VM for every cluster. This increases resource usage quickly. If a cloud system had 100 clusters then there would be 100 VMs that would only be used for monitoring. It would be beneficial if it was possible to have a smaller number of VMs used to scan multiple clusters. However the current system is unable to perform this type of task.

Currently Slack is used as a medium to communicate between the administrators and the scanning mechanism. This has proven to be an interesting tool allowing for bidirectional communication and quick information retrieval. However the reliance on Slack may be jeopardized if Slack decides to change their API, limit bots, or goes out of business. Fortunately the system was designed in such a way to allow any communication medium to be added to the system. Therefore if Slack does not continue to meet the standards that an organization needs another tool can be used to replace Slack with ease. Some alternatives to Slack could be Stride [17] or IRC.

An issue that was ran into later during the creation of the project was that the Slack channel would get cluttered if a lot of different devices were miss-configured. Every VM that does not follow the configuration of the scanning tool will create an error. One way to sort out a large amount of created data in the Slack channel would be to add some type of symbols to differentiate between low and high risk problems. These symbols would be able to indicate the severity level / risk that each output would pose. This would allow an administrator to visually differentiate a high risk issue that needed his attention over some low risk issues that could be resolved at a later date. The severity level would have to be decided by the administrative team. Some ideas to fix this were thought of but were not implemented due to time constraints.

[17]https://www.stride.com/

14 Future Work

The project has a large potential for expansion. As the code written now is only a prototype there is more work that can be added to expand the tools usefulness. As the code is created with modularity in mind, a user can easily add new scans into the project. If a user wanted, for example, to ensure that a website is correctly configured they could add the nikto [13] vulnerability scanner. The scan can then be run and the data would be added to the database. Another future work that could be expanded on is allowing for different methods of interacting with the system. Currently Slack is used, however administrators might want email reports or an IRC channel instead of Slack. Fortunately this can easily be integrated and several systems can be used asynchronously. This allows for administrators to custom configure this system to their desired needs and specifications. In addition the project is opensource, therefore anyone is allowed to take the code and modify it. This allows for individuals to expand the project in whatever direction that they see fit.

15 Conclusion

As was shown the cloud environment is a complex system that is steadily growing and innovating. These cloud resources give users an powerful and flexible tool to create complex systems that are independent from hardware. As users gain full controls of systems directly connected to the Internet, system security becomes user based, however as mentioned users may not know best practices therefore leaving systems vulnerable to compromise. With users and organizations switching over to using cloud resources in everyday operations new attacks become ever more present. Many of the leaders in the public cloud have created proprietary solutions to attempt to protect their users' and systems from compromise. However many of the private clouds using open source solutions, like OpenStack, may not have the resources or knowledge to create such solutions. Therefore this project was designed and created to prototype a system that is able to actively scan private clouds for potential vulnerabilities. Administrators are able to see results and query an automated system for further information with ease. This thesis provided a great learning opportunity about the threat landscape of cloud systems and how a modular system can be designed. It is the hope of the author that this system can be used by other people to ensure that any private cloud environment is secure.

References

[1] Cloud history: Timeline of cloud computing. `http://sourcedigit.com/497-timeline-history-of-cloud-computing`.

[2] Cloud security amazon web services (AWS). `https://aws.amazon.com/security`. Accessed: 2017-04-14.

[3] The FreeBSD project. `https://www.freebsd.org`. Accessed: 2017-04-10.

[4] Google trends for cloud computing. `https://trends.google.com/trends/explore?date=all&q=%2Fm%2F02y_9m3`. Accessed: 2017-03-04.

[5] A hackable text editor for the 21st century. `https://atom.io`. Accessed: 2017-04-04.

[6] The linux project homepage. `https://www.linuxfoundation.org`. Accessed: 2017-04-04.

[7] OpenStack website. `https://www.openstack.org`. Accessed: 2017-02-02.

[8] Friendly university network probing. `https://www.ch.cam.ac.uk/computing/policy-friendly-probing`, 2017. Accessed: 2017-04-04.

[9] Nessus vulnerability scanning. `https://www.tenable.com/products/nessus/nessus-professional`, 2017. Accessed: 2017-12-14.

[10] OpenStack heat platform. `https://wiki.openstack.org/wiki/Heat`, 2017. Accessed: 2017-03-12.

[11] Openvas vulnerability scanning. `http://www.openvas.org/`, 2017. Accessed: 2017-12-14.

[12] Volitility framwork. `http://www.volatilityfoundation.org/`, 2017.

[13] Website vulnerability scanner. `https://github.com/sullo/nikto`, 2017. Accessed: 2017-04-04.

[14] Adobe. Photoshop cc. `https://www.adobe.com/products/photoshop.html`, January 2018.

[15] A Amini, N Jamil, AR Ahmad, and MR Z'aba. Threat modeling approaches for securing cloud computing. *Journal of Applied Sciences*, 15(7):953, 2015.

[16] Amazon AWS. `https://aws.amazon.com`, 2017. Accessed: 2017-03-11.

[17] Microsoft Azure. `https://azure.microsoft.com/en-us`, 2017. Accessed: 2017-04-01.

[18] Michael Bauer. *Linux server security.* " O'Reilly Media, Inc.", 2005.

[19] Neil Briscoe. Understanding the osi 7-layer model. *PC Network Advisor*, 120(2), 2000.

[20] Catalin Cimpanu. Honeypot server gets infected with wannacry ransomware 6 times in 90 minutes. `https://www.bleepingcomputer.com/news/security/honeypot-server-gets-infected-with-wannacry-ransomware-6-times-in-90-minutes/`, May 2017.

[21] Google Cloud Compute. `https://cloud.google.com`, 2017. Accessed: 2017-03-14.

[22] Evan Cooke, Farnam Jahanian, and Danny McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. *Steps to Reducing Unwanted Traffic on the Internet Workshop*, 5:6–6, 2005.

[23] Jim Curry. Introducing OpenStack. `https://www.openstack.org/blog/2010/07/introducing-openstack/`, 2010. Accessed: 2017-05-04.

[24] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: Issues and challenges. *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference.*

[25] Justin Ellingwood. Securing a server. `https://www.digitalocean.com/community/tutorials/7-security-measures-to-protect-your-servers`, 2017. Accessed: 2017-04-02.

[26] Justin Ellingwood. 7 security measures to protect your servers. `https://www.digitalocean.com/community/tutorials/7-security-measures-to-protect-your-servers`, January 2018.

[27] The Apache Software Foundation. Apache. `https://httpd.apache.org/`, January 2018.

[28] Google. Gmail. `http://gmail.com/`, January 2018.

[29] IRC. Irc. `http://www.irc.org/`, January 2018.

[30] Tom Kellerman. Cyber-threat proliferation: Today's truly pervasive global epidemic. *IEEE Security Privacy*, 8(3):70–73, 2010.

[31] Derrick Kondo, Bahman Javadi, Paul Malecot, Franck Cappello, and David P Anderson. Cost-benefit analysis of cloud computing versus desktop grids. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–12. IEEE, May 2009. https://ieeexplore.ieee.org/document/5160911/.

[32] Ronald L Krutz and Russell Dean Vines. *Cloud security: A comprehensive guide to secure cloud computing.* Wiley Publishing, 2010.

[33] Dan Lambright. Unobtrusive intrusion detection in OpenStack. `https://www.openstack.org/summit/vancouver-2015/summit-videos/presentation/unobtrusive-intrusion-detection-in-openstack`, 2017. Accessed: 2017-02-20.

[34] Bill Loeffler. What is infrastructure as a service? `https://social.technet.microsoft.com/wiki/contents/articles/4633.what-is-infrastructure-as-a-service.aspx`. Accessed: 2017-03-20.

[35] Microsoft. Office 365. `https://www.office.com/`, January 2018.

[36] Ankur Mishra, Ruchita Mathur, Shishir Jain, and Jitendra Singh Rathore. Cloud computing security. *International Journal on Recent and Innovation Trends in Computing and Communication*, 1(1):36–39, 2013.

[37] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 2013.

[38] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 36(1):42–57, 2013.

[39] mongodb. mongodb. `https://www.mongodb.com/`, January 2018.

[40] MySQL. Mysql. `https://www.mysql.com/`, January 2018.

[41] Paulo Neto. Demystifying cloud computing. *Proceeding of Doctoral Symposium on Informatics Engineering*, 2011. Universidade do Porto. https://paginas.fe.up.pt/ prodei/dsie11/images/pdfs/s6-3.pdf.

[42] Lily Hay Newman. The biggest cybersecurity disasters of 2017 so far. `https://www.wired.com/story/2017-biggest-hacks-so-far/`, June 2017.

[43] Boon-Yuen Ng, Atreyi Kankanhalli, and Yunjie Calvin Xu. Studying users' computer security behavior: A health belief perspective. *Decision Support Systems*, 46(4):815–825, 2009.

[44] Digital Ocean. `https://www.digitalocean.com`, 2017. Accessed: 2017-04-01.

[45] Ahmed Patel, Mona Taghavi, Kaveh Bakhtiyari, and Joaquim Celestino Nior. An intrusion detection and prevention system in cloud computing: A systematic review. *Journal of network and computer applications*, 36(1):25–41, 2013.

[46] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio Martín Llorente, Rubén Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4–1, 2009.

[47] Slack. Slack. `https://slack.com/`, January 2018.

[48] Jeffrey M Stanton, Kathryn R Stam, Paul Mastrangelo, and Jeffrey Jolton. Analysis of end user security behaviors. *Computers security*, 24(2):124–133, 2005.

[49] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, dec 2008.

[50] Seth Webster, Richard Lippmann, and Marc Zissman. Experience using active and passive mapping for network situational awareness. In *Network Computing and Applications, 2006. NCA 2006. Fifth IEEE International Symposium on*, pages 19–26. IEEE, 2006.

[51] WordPress.org. Wordpress. `https://wordpress.org/`, January 2018.

[52] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Generation computer systems*, 28(3):583–592, 2012.

User recommendations to secure virtual machines in OpenStack

Securing access to the Virtual Machine

One of the easiest ways for an advisory to get access to a virtual machine in the cloud is through poorly configured login credentials. there are bots on the web that will try to brute force login username and passwords on exposed ssh interfaces. It is therefore important for all machines to use strong login methods. One of these methods is to use a public private key pair instead of a password. Bellow are the instrustions on how to set up an ssh keypair.

1. Ensure that one has created an ssh key pair.

- In a terminal run the following command:

```
$ ssh-keygen -t rsa -b 4096
# The `ssh-keygen` command will create the key
# The `-t` option will set the cryptosystem to rsa
# The `-b` will set the key length to 4096 bits
Generating public/private rsa key pair.
Enter file in which to save the key (/home/groot/.ssh/id_
rsa) #Your path goes here
```

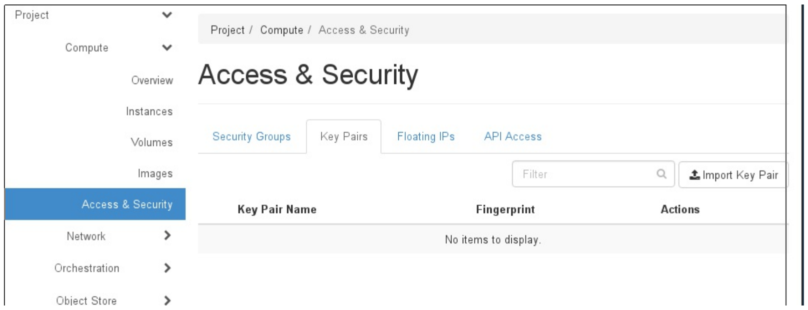
```
Enter passphrase (empty for no passphrase) #Enter a passw
ord
Enter same passphrase again: #Reenter your password
...
```

- You now will have a public and private key pair in the location that you specified. You can then submit your public key (`.pub`) when you try and `ssh` into a machine.

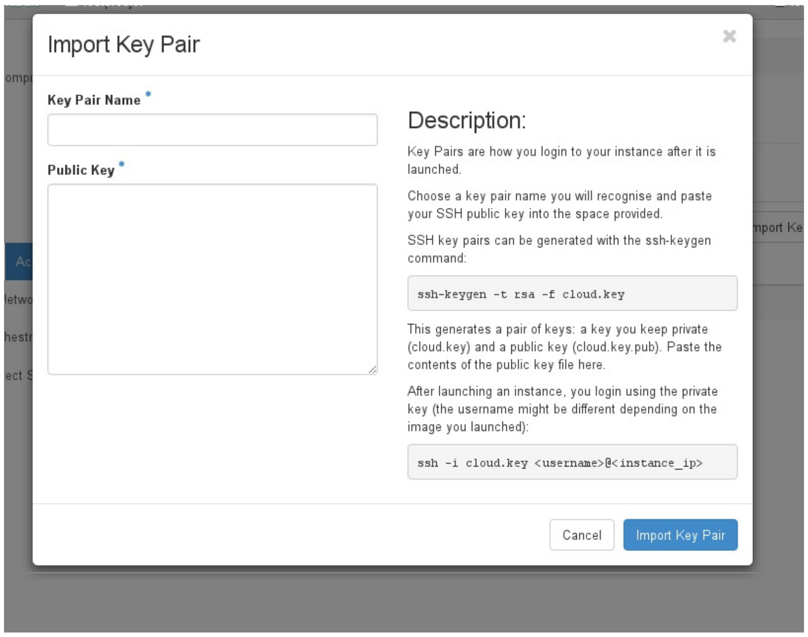
```
$ ssh -i #Path to your private key# username@machine
```

2. You must upload your public key into OpenStack control panel:

- Login to the OpenStack administration panel and select "Access & Security" in the left hand column.
- Next select the "key Pair" located to bellow the "Access & Security" title.



- Once on the page proceed to click on the "Import Key Pair" button located at the top right of the screen.
- A pop-up should appear that looks similar to the following:



Description:

Key Pairs are how you login to your instance after it is launched.

Choose a key pair name you will recognise and paste your SSH public key into the space provided.

SSH key pairs can be generated with the ssh-keygen command:

```
ssh-keygen -t rsa -f cloud.key
```

This generates a pair of keys: a key you keep private (cloud.key) and a public key (cloud.key.pub). Paste the contents of the public key file here.

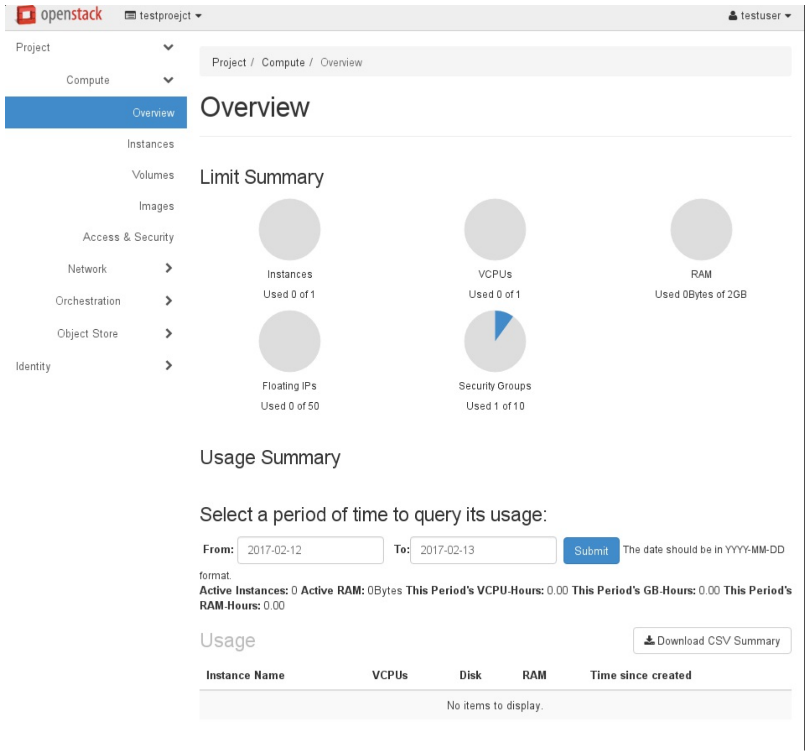
After launching an instance, you login using the private key (the username might be different depending on the image you launched):

```
ssh -i cloud.key <username>@<instance_ip>
```

- Go back to your terminal and find your public key (`.pub`). Copy the contents and paste them into the field marked “Public Key”. Then add a name to your key and click the “Import Key Pair” button at the bottom of the pop-up.

Creating a new machine

1. Access the main control panel for OpenStack:



Overview

Limit Summary

Instances	VCPUs	RAM
Used 0 of 1	Used 0 of 1	Used 0Bytes of 2GB

Floating IPs	Security Groups
Used 0 of 50	Used 1 of 10

Usage Summary

Select a period of time to query its usage:

From: 2017-02-12 **To:** 2017-02-13 Submit The date should be in YYYY-MM-DD format.

Active Instances: 0 **Active RAM:** 0Bytes **This Period's VCPU-Hours:** 0.00 **This Period's GB-Hours:** 0.00 **This Period's RAM-Hours:** 0.00

Usage

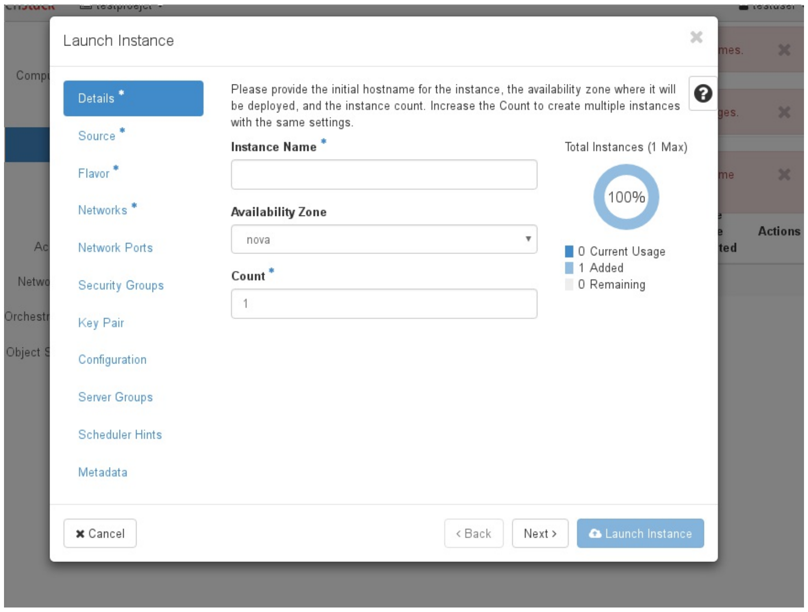
Download CSV Summary

Instance Name	VCPUs	Disk	RAM	Time since created
No items to display.				

2. Select the “Instance” tab on the left hand side of the screen.

3. Now select the “Launch Instance” button on the upper right hand side of the screen.

4. A pop-up should appear that look similar to the following:



Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name *

Availability Zone nova

Count * 1

Total Instances (1 Max)

100%

0 Current Usage
1 Added
0 Remaining

5. Fill out the necessary setting.

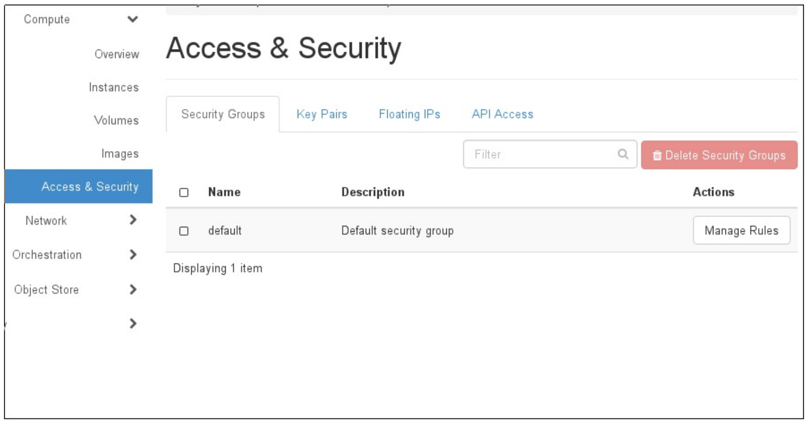
- Ensure that the “Security Group” tab has the correct group assigned to the VM.
- Ensure that the correct public key is selected in the “Key Pair” tab

Security Rules

Security rules are a set of instructions that can be placed on to VMs that will limit that particular VMs ability to access external environments. They act very similar to a firewall with allowing or blocking different types of connections to the individual VMs. To set up proper rules the following steps should be taken:

1. Log into OpenStack and select the “Access & Security” tab on the left hand side.

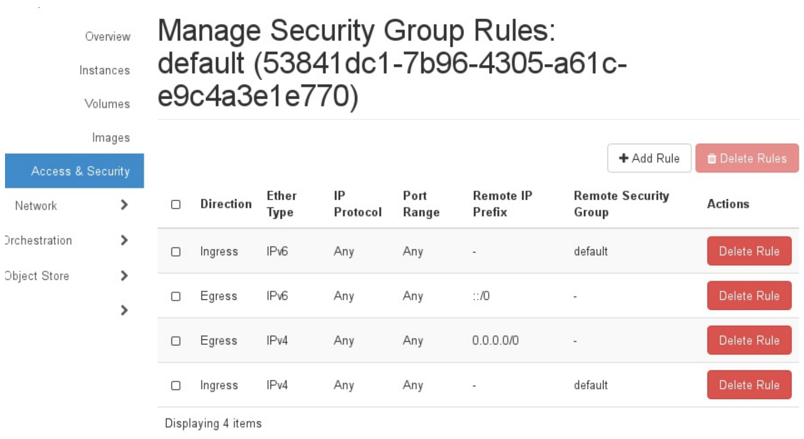
2. You will now see a list of security groups that you are part of. Select the group that you would like to edit by clicking the “Manage Rules” button on the right hand side.



	Name	Description	Actions
☐	default	Default security group	Manage Rules

Displaying 1 item

3. The new screen should look something like this:



Manage Security Group Rules: default (53841dc1-7b96-4305-a61c-e9c4a3e1e770)

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
Ingress	IPv6	Any	Any	-	default	Delete Rule
Egress	IPv6	Any	Any	::/0	-	Delete Rule
Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
Ingress	IPv4	Any	Any	-	default	Delete Rule

Displaying 4 items

Lets break down what the different components are:

- The “Direction” field describes if inbound or outbound traffic is affected by the rule
- “Ether Type” describes wether the rule talks about IPv4 or IPv6 traffic
- “IP Protocol” describes what type of traffic is permitted or blocked (i.e. http, pop3, etc.)
- “Port Range” what ports that rule applies to for that machine
- “Remote IP Prefix” describes what IP address the traffic should come from

As with any computer system accessible to the internet great lengths should be taken to ensure that a machine can not be compromised. As cloud based solutions often see machines turning on and off frequently a strong set of “Security Rules” can help ensure a healthy virtual environment. Bellow are a set of recommendations to follow:

- Only allow port 22 (ssh) and only the ssh protocol to be allowed to connect to VMs from remote hosts.
- If a machine or group of machine dont need access the the internet (i.e. database machine) ensure that the security group only allows internal network traffic to come from and to that machine.
- Open only the necessary port for specific machines to access the internet.
- If feasible block all outgoing traffic (ensure that one does not block ssh if needed to connect from the outside) and then open only the ports that are needed.

A good reference point for security rules are Linux iptable rules (http://netfilter.org/).